НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Геометричне моделювання в інформаційних системах»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Модуль формування базових даних інформаційної системи автоматизованої підтримки навчальної діяльності кафедри**»**

Виконав:
студент IV курсу, групи ТР-61
Белень Олександр Михайлович _____

Керівник:
Професор, доктор технічних наук, доцент
Шушура Олексій Миколайович _____

Рецензент:
Доктор технічних наук, професор
Сторчак Камілла Павлівна _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ

на дипломну роботу студенту

Беленю Олександру Михайловичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Модуль формування базових даних інформаційної системи автоматизованої підтримки навчальної діяльності кафедри_____

керівник роботи _Шушура Олексій Миколайович, д.т.н., доцент_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від “25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи “” червня 2020 року_____

3. Вихідні дані до роботи тимчасове положення про організацію освітнього процесу в КПІ ім. Ігоря Сікорського_____
_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)_____

1) постановка задачі;_____

2) огляд інформаційних систем автоматизованої підтримки навчальної діяльності;_____

3) проектування системи автоматизованої підтримки навчальної діяльності;_____

4) Програмна реалізація системи автоматизованої підтримки навчальної діяльності_____

5) Методика роботи користувача з програмною системою

_____
_____
_____

5. Перелік ілюстративного матеріалу
Мета, постановка задачі, діаграми прецедентів, діаграма діяльності, діаграма розгортання, архітектура системи, концептуальна модель бази даних, вибір засобів розробки, приклади інтерфейсу, висновки.
_____
_____

6. Дата видачі завдання ”___”_____2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної робот	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	11.10.2019	
2.	Вивчення та аналіз задачі	14.10.2019 – 27.12.2019	
3.	Розробка архітектури та загальної структури системи	03.02.2020 – 06.03.2020	
4.	Розробка структур окремих підсистем	09.03.2020 – 10.04.2020	
5.	Програмна реалізація системи	13.04.2020 – 15.05.2020	
6.	Оформлення пояснювальної записки	18.05.2020 – 05.06.2020	
7.	Захист програмного продукту	10.06.2020	
8.	Передзахист	10.06.2020	
9.	Захист	17.06.2020	

Студент _____ _____Белень О. М._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Шушура О. М._____
(підпис) (прізвище та ініціали,)

Анотація

Метою даної дипломної роботи є розробка модулю формування базових даних інформаційної системи автоматизованої підтримки навчальної діяльності кафедри. Розроблений модуль надає можливість вводити та редагувати базові дані, необхідні для роботи системи. До цієї інформації відносяться дані щодо структури університету, спеціальностей, викладацького складу та інші. Спроектована база даних, обрані засоби реалізації та розроблене програмне забезпечення.

Дипломну роботу виконано на 90 аркушах, вона містить 41 ілюстрацію, 7 таблиць, 30 посилань на літературу та 4 додатки.

Abstract

The purpose of this thesis is to develop a module for the formation of basic data of the information system of automated support of educational activities of the department. The developed module provides the ability to enter and edit the basic data required for the system. This information includes data on the structure of the university, specialties, teaching staff and others. The database is designed, implementation tools are selected and software is developed.

The thesis is completed on 90 sheets, it contains 41 illustrations, 7 tables, 30 references to literature and 4 annexes.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ПОЗНАЧЕНЬ

CSS – Cascading Style Sheets - Каскадні таблиці стилів

EF – Entity Framework

HTML – HyperText Markup Language – мова розмітки веб-документів

ORM – Object-relational mapping – об’єктно-реляційна проекція

LINQ – Language Integrated Query – запити, інтегровані в мову

SQL – Structured query language — мова структурованих запитів

SSMS – SQL Server Management Studio

БД – база даних

ВНЗ – вищий навчальний заклад

ОС – операційна система

СУБД – система управління базами даними

ВСТУП

Стрімкий розвиток комп'ютерної техніки, застосування її в усіх галузях, повсякденному житті, можливість сучасного комп'ютера зберігати, представляти та обробляти будь-яку інформацію, зумовило їх використання і в освітній діяльності, де вже стало традиційним використовувати інформаційні системи як для адміністративних задач і управління закладами освіти, так і для безпосереднього використання в освітньому процесі. Починаючи з 80-х років минулого сторіччя спостерігається стійка тенденція щодо використання найпередовіших досягнень в галузі інформаційних технологій, насамперед, в області освіти.

Функціонування сучасного університету неможливе без використання різних інформаційних та телекомунікаційних систем автоматизуючих різні процеси вишу. Зокрема, без таких систем не можна уявити роботу бухгалтерії. В подальшому ефективність використання таких інформаційних систем почала використовуватися відділами кадрів, факультетами та кафедрами.

Тим не менше, до цих пір різні підрозділи університетів часто використовують часткові, не пов'язані і навіть несумісні між собою програмні рішення. Однак, передовими університетами світу визнано, що ефективне вирішення задач щодо керування сучасним університетом неможливе без єдиної інтегрованої інформаційно-аналітичної системи.

Звісно ж, у кожному навчальному закладі є люди, які відповідають за організацію навчального процесу, але дана робота є складною. У зв'язку з цим дані системи мають наступні завдання:

- інтеграція ресурсів у інформаційний простір;
- можливість оперативного обміну науковою та навчальною інформацією;
- доступ до інформаційних баз даних та знань;

– ефективне формування та збереження власних інформаційних ресурсів, підготовку інформаційних послуг;
– перехід на електронний документообіг;
– автоматизація всіх технологічних процесів бібліотеки;
– забезпечення безпеки персоналу;
– формування розкладу для студентів та викладачів.

Сьогодні провідні університети вважають своєю місією, поряд з завданнями бути центрами культури, науки й освіти в регіоні та Україні, формувати прогресивну, демократичну, гуманістичну, соціально-політичну думку молоді, виховувати гармонійно розвинену особистість, патріота української держави, ще й здійснювати підготовку висококваліфікованих конкурентоздатних фахівців для вирішення комплексу проблем розвитку, в першу чергу, національної та світової економіки. Реалізація нової місії в суспільстві під силу тільки такому університету, який сам змінюється, трансформується відповідно до нових умов.

Отже, університет як інституційна система, та його інформаційно-комунікаційна інфраструктура вкотре, в умовах переходу від індустріального до інформаційного суспільства, зазнає суттєвих змін. Під час реформування сучасної вищої освіти на перший план виходить завдання інтегрування інформаційних та телекомунікаційних технологій в освітній процес та в управлінні освітою.

1 ПОСТАНОВКА ЗАДАЧІ

У сучасному світі в останній період часу сучасні технології швидко розвиваються, є поширеним і доступним, майже всюди, Інтернет-покриття, що відкриває унікальні можливості для освіти. Завдяки йому сучасний студент має змогу на для доступу до будь-якої інформації, необмежену можливість одержання знань, покращувати свою інтелектуальну діяльність та вдосконалюватися.

Через ці можливості створюються системи для університетів для покращення навчальної діяльності. Також на даний момент не існує єдиного шаблону для реалізації даної системи.

Модуль формування базових даних є одним з головних компонентів такої системи, адже всі інші модулі, які в майбутньому будуть в системі, будуть використовувати саме ці дані, які були сформовані. Доцільно спроектувати одну єдину базу даних, яка зберігатиме всю потрібну інформацію. Модуль формування базових даних дасть можливість систематизувати заповнення таблиць даними, що потрібні для функціонування кафедри.

Мета дипломної роботи – це створити модуль формування базових даних для інформаційної системи кафедри, що дасть змогу в подальшому заповнювати та керувати даними, а при розробці додаткових спеціалізованих модулів забезпечить можливість їх використання.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- провести огляд та аналіз існуючих інформаційних систем підтримки навчальної діяльності;
- спроектувати архітектуру системи та розробити структуру її бази даних;

- вибрати засоби розробки та виконати програмну реалізацію системи;
- розробити опис програми та методику її використання для користувача.

Модуль повинен забезпечити виконання наступних функцій, як перегляд, редагування та видалення даних з бази. Також система повинна надавати функціонал, який буде розподілятися за ролями, які будуть доступні в ній.

Важливою можливістю у роботі системи має бути обмеження прав певних ролей користувачів. Наприклад, працівник навчального відділу має змогу на додавання та редагування даних, а працівник відділу навчальних планів може тільки використовувати ці дані для своєї роботи.

Також після вирішення всіх поставлених задач застосунок повинен відповідати таким вимогам:

- мати зручний та зрозумілий інтерфейс користувача;
- бути доступною для додавання нового функціоналу;
- працювати на різних ОС;
- використовувати веб-інтерфейс;
- мати добре сформоване сховище даних для подальшої роботи з системою та додавання нових модулів.

2 ОГЛЯД ІНФОРМАЦІЙНИХ СИСТЕМ ПІДТРИМКИ НАВЧАЛЬНОЇ ДІЯЛЬНОСТІ

Інформаційні технології стали невід'ємною частиною сучасного життя. Вони назавжди змінили наш світ бізнесу, культури, освіти та виробництва. Водночас постала проблема необхідності обробки великої кількості інформації, а отже, створення таких систем, що будуть полегшувати навчальний процес та економити час. Освіта має орієнтуватися на технологіях, які повинні формувати в учнів уміння вчитися та пристосовуватися до потреб ринку праці, а також полегшувати викладач навчальний процес, оскільки їх робота є недооціненою та досить стресовою.

Навчальний процес у вищому навчальному закладі достатньо важкий, і його ефективна організація потребує значних зусиль. Ним можна назвати систему організаційних та дидактичних заходів, спрямованих на реалізацію змісту освіти на певному освітньому або кваліфікаційному рівні відповідно до державних стандартів освіти. *Навчальний процес повинен* організовуватися з урахуванням можливостей сучасних інформаційних технологій навчання та орієнтується на формування освіченої, гармонійно розвиненої особистості, здатної до постійного оновлення наукових знань, професійної мобільності та швидкої адаптації до змін і розвитку в соціально-культурній сфері. Один з ключових факторів організації навчального процесу – це складання розкладу занять, яке повинно забезпечувати рівномірне навантаження як для студентів, так і для викладачів, рівномірно використовувати аудиторні ресурси з урахуванням специфіки різних приміщень(лабораторії, потокові аудиторії, аудиторії зі спеціальним технічним обладнанням та інші), враховувати побажання викладачів та інше.

Основною метою розроблення та впровадження даних систем є підвищення якості освітнього процесу. Цієї мети можна досягти завдяки постійному моніторингу параметрів якості, забезпеченню достовірності й швидкості отримання інформації щодо різних аспектів організації навчального процесу з необхідним ступенем деталізації і, як наслідок, обґрунтованості й оперативності ухвалення управлінських рішень, що безпосередньо впливають на виконання університетом освітньої функції.

Вивчення наявних систем керування й аналітичної літератури щодо цього питання показало, що у наш час в нашій країні не існує універсальних програмних продуктів, призначених для підтримки навчальної діяльності університетів, факультетів та кафедр. Наявні ж системи в багато чому не підходять для вирішення конкретних задач конкретного університету.

2.1 Огляд існуючих систем для університетів

В наш час існує велика кількість автоматизованих систем підтримки навчальної діяльності в університеті, кожна з яких має свої переваги та недоліки. Тому задля оцінення та покращення рішення необхідно ознайомитися з тими рішеннями, які є доступними для користувачів в даний момент часу. Розглянемо деякі з них та проведемо порівняння.

Інформаційно-телекомунікаційна система «Електронний кампус» - це прикладне програмне забезпечення, яке є елементом інформаційно-телекомунікаційного середовища університету та використовується для інформаційної підтримки повсякденної діяльності студентів, викладачів, співробітників університету, а так само для інформаційної підтримки всіх видів інноваційної діяльності в університеті.

Електронний кампус об’єднує внутрішні інформаційні ресурси (навчальні, методичні та інші), надає єдиний доступ до них на основі єдиних системних і технологічних рішень та забезпечує їх використання для ефективного управління та планування науково-освітнім процесом.

Платформа «Електронний кампус» є безкоштовною та доступна тільки для студентів НТУУ «КПІ». Дана система забезпечує:

- інформаційну підтримку освітнього процесу кафедр університету шляхом надання віртуальних кабінетів за профілями користувачів: студент, викладач-науковець, куратор, методист кафедри;
- розповсюдження інформації про майбутні події і заходи в рамках освітнього процесу комунікації між його учасниками;
- використання студентами закритого сховища навчально-методичного забезпечення навчального процесу з можливістю відкриття доступу всім бажаючим до визначених навчально-методичних матеріалів.

В системі розроблені віртуальні кабінети за профілями користувачів: студент, викладач-науковець, методист кафедри.

Одним з недоліків системи є відсутність розкладу занять як для викладачів, так і для студентів. Іншим недоліком є відсутніх деяких даних, які можуть цікавити користувача. Також великим мінусом системи, я вважаю, обмеження у завантаженні файлів розмір яких перевищує 2 МБ.

Інтерфейс вигляду системи зображено на рисунку 1.1.

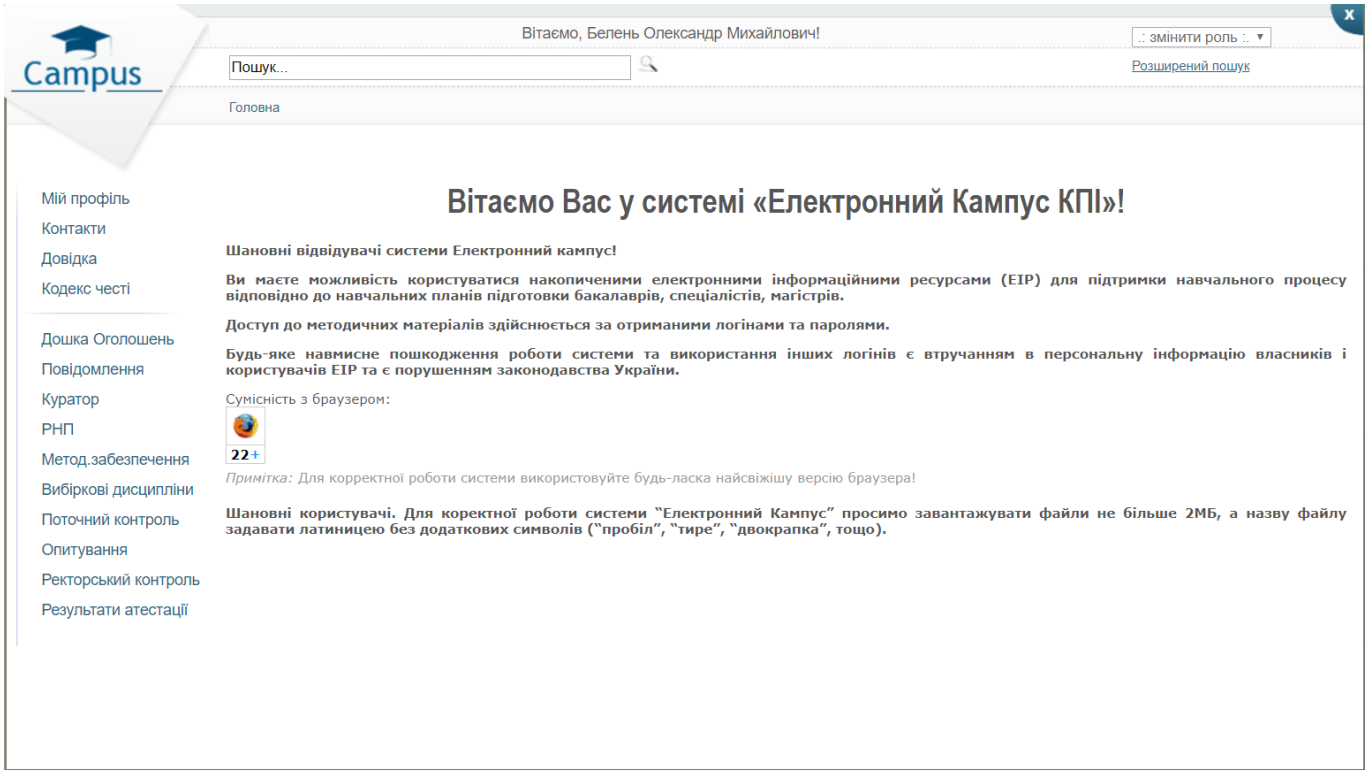

Рисунок 1.1 – Інтерфейс системи «Електронний кампус»

Також досить популярною системою в Україні є програма «Курс: ВНЗ», яка враховує відомості про інфраструктуру навчального закладу, а саме корпуси, приміщення, поверхи та інші, адміністрацію та викладачів закладу, студентів та дисципліни та багато іншої інформації пов'язаної з роботою та діяльністю університету. Вікно системи зображено на рисунку 2.2.

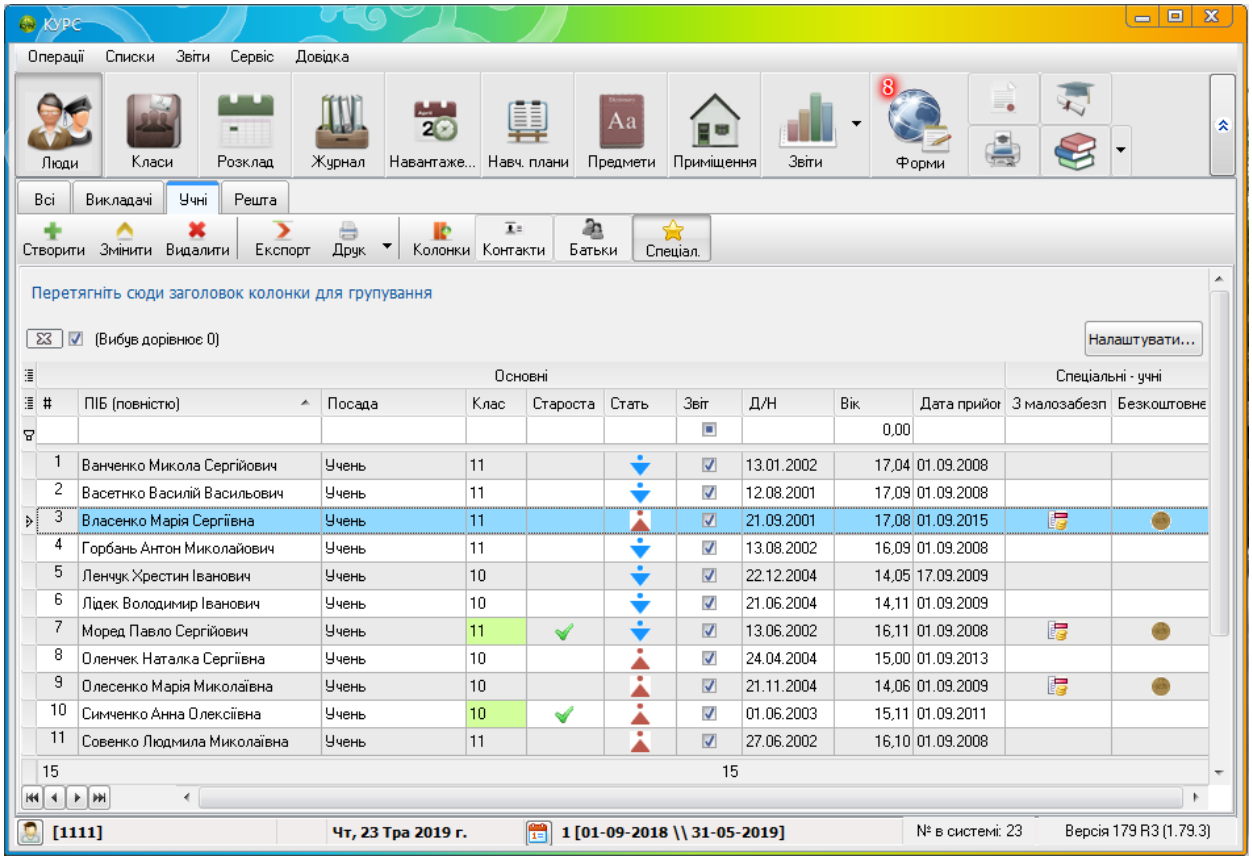

Рисунок 2.2 – Вікно програми «Курс: ВНЗ»

Комплексна інформаційна система «Університет» - це відкрите модульне рішення для автоматизації діяльності ВНЗ.

Система має модульну структуру, яка дозволяє використовувати як весь функціонал програми, так і окремо необхідні для користування модулі. Така система призначена для автоматизації найбільш значущих робочих процесів вишу. Використання сучасних технологій зберігання, обробки та аналізу даних дозволяє перейти на новий якісний рівень процеси керування вищим навчальним закладом.

Система також дозволяє здійснювати моніторинг якості організації навчального процесу, якості підготовки спеціалістів, ведення наукової діяльності, ефективності системи керування. В швидкому доступі керівника з’являються всі необхідні показники підрозділу, сповіщення про здійснені операції.

Вікно інтерфейсу платформи зображено на рисунку 1.2.

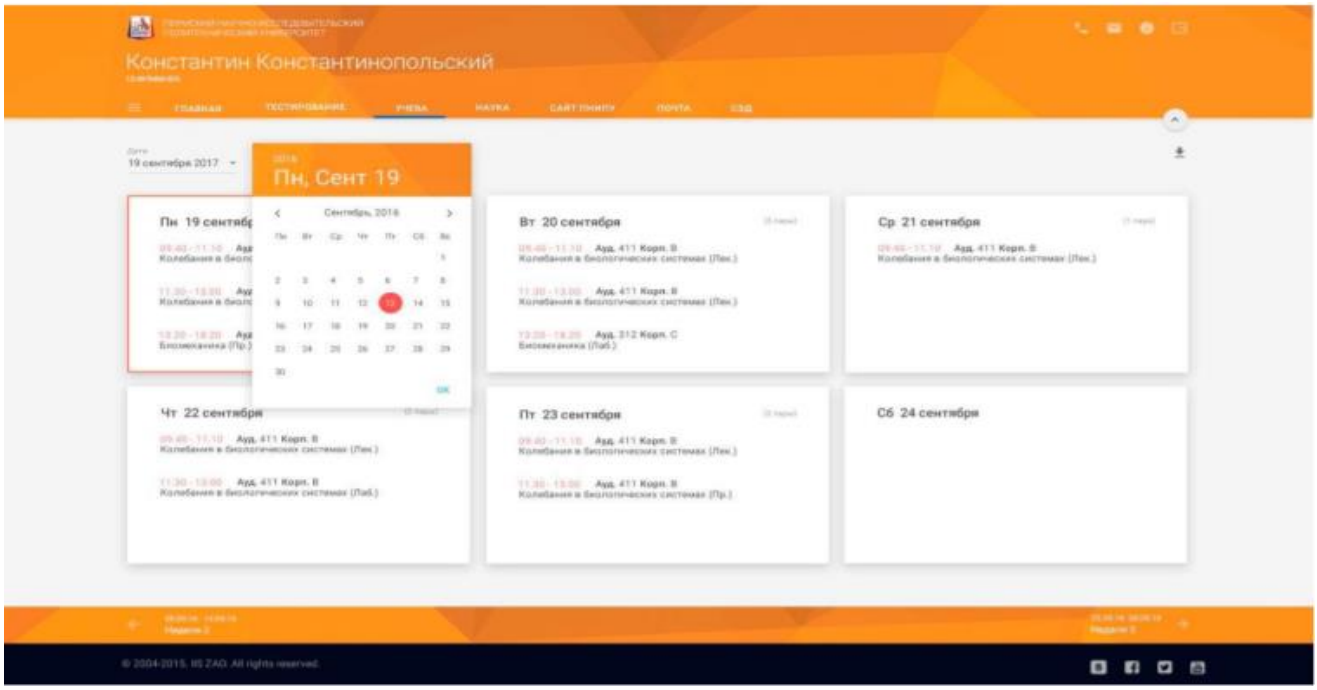

Рисунок 1.2 – Вікно інтерфейсу платформи «Університет»

Система «Студент Львівської політехніки» дозволяє створювати віртуальний кабінет студента, в якому він має наступні можливості:

– перегляд новин університету та важливої інформації щодо студентів;
– посилання на різні університетські онлайн ресурси та університетські сервіси;
– формування власного навчального плану та актуальні пропозиції вибору дисциплін;
– збереження особистих даних студента.

Система схожа до «Електронного кампусу», тільки має дещо менший функціонал. Доступна вона ж тільки для студентів Львівської політехніки.

Недоліками ж даної системи є малий функціонал, а також відсутність розкладу для користувачів. Інтерфейс головної сторінки зображено на рисунку 1.3.

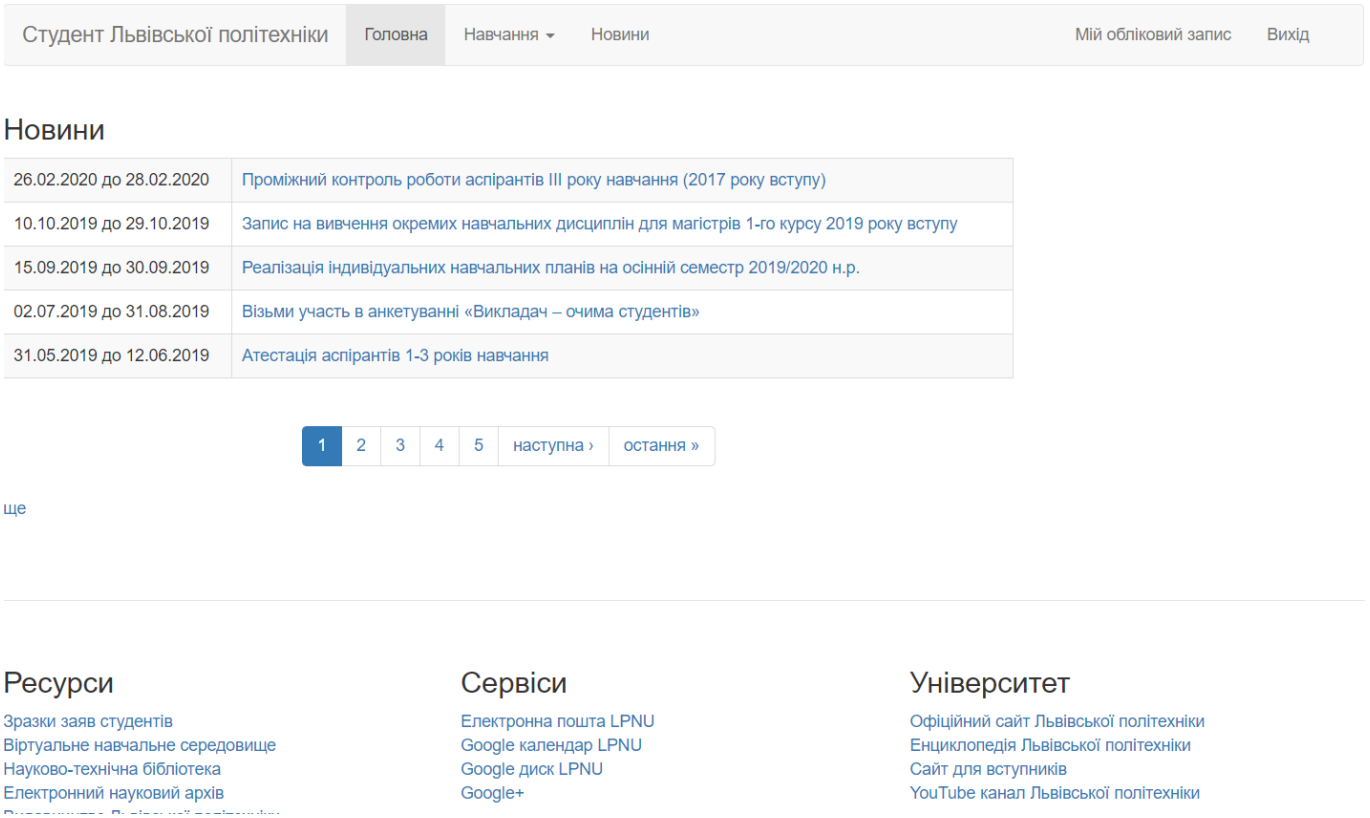

Рисунок 1.3 – Вікно інтерфейсу платформи «Студент Львівської політехніки»

Магелан – ця система є досить популярною в країнах колишнього СРСР. Вона має гнучку архітектуру та простий інтерфейс, що дозволяє одразу перейти до експлуатації системи. Система постійно оновлюється та підтримується, щоб уникнути різним програмним проблемам. В базі системи зберігається інформація по всьому навчальному закладі та його начальних процесах. Програма включає в себе 17 різних модулів, особисті кабінети студентів, викладача, а також абітурієнта. З мінусів системи це те, що її ціна занадто велика і не відповідає функціоналу системи. Також недоліком є інтуїтивно незрозумілий інтерфейс. Користувачу потрібно спочатку витратити декілька годин, щоб розібратися з системою.

Вікно інтерфейсу програми зображено на рисунку 1.4.

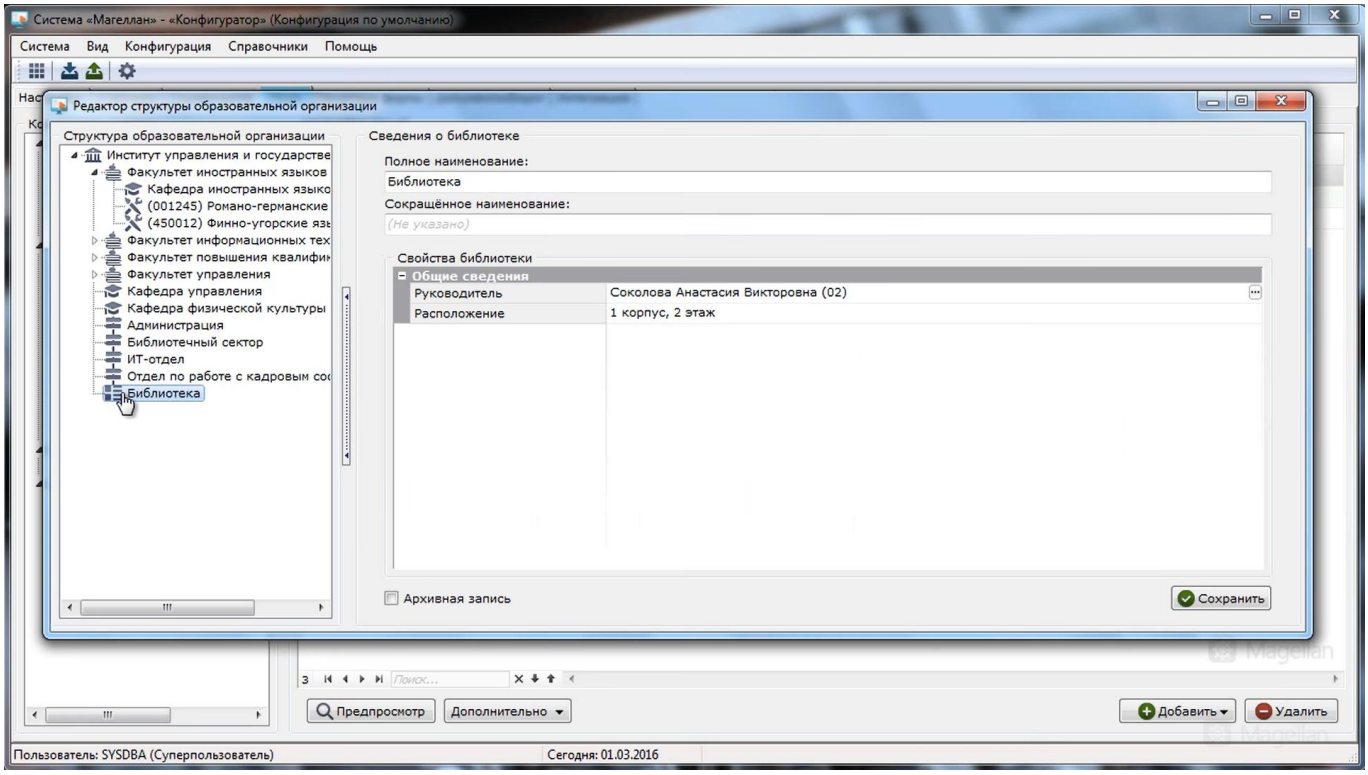

Рисунок 1.4 – Вікно інтерфейсу системи «Магелан»

Також гарним зразком є комплексне рішення ODEMS (англ. Optimal Digital Education Management System). Рішення представляє собою повне автоматизоване програмне забезпечення для управління освітою, підходить для будь-якого навчального закладу. Система надає широкий спектр можливостей (Рис. 1.5), які можна налаштовувати окремо для кожного процесу.

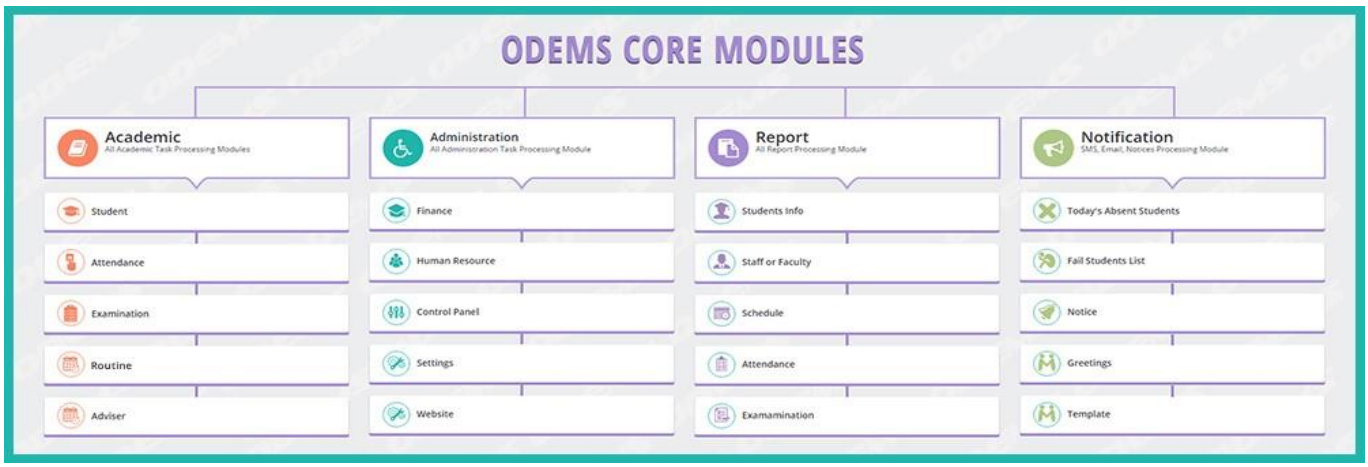

Рисунок 1.5 – Схема роботи ODEMS

Основною перевагою ODEMS є те що здебільшого всі дані зберігаються в «хмарному» сховищі, а тому доступні користувачам у будь-який момент та в будь-якому місці. Також система має окремий модуль майже для кожної задачі (див. рис. 1.6), наприклад, студентський модуль для управління обліковими записами студентів та академічний модуль для обліку присутності та успішності.

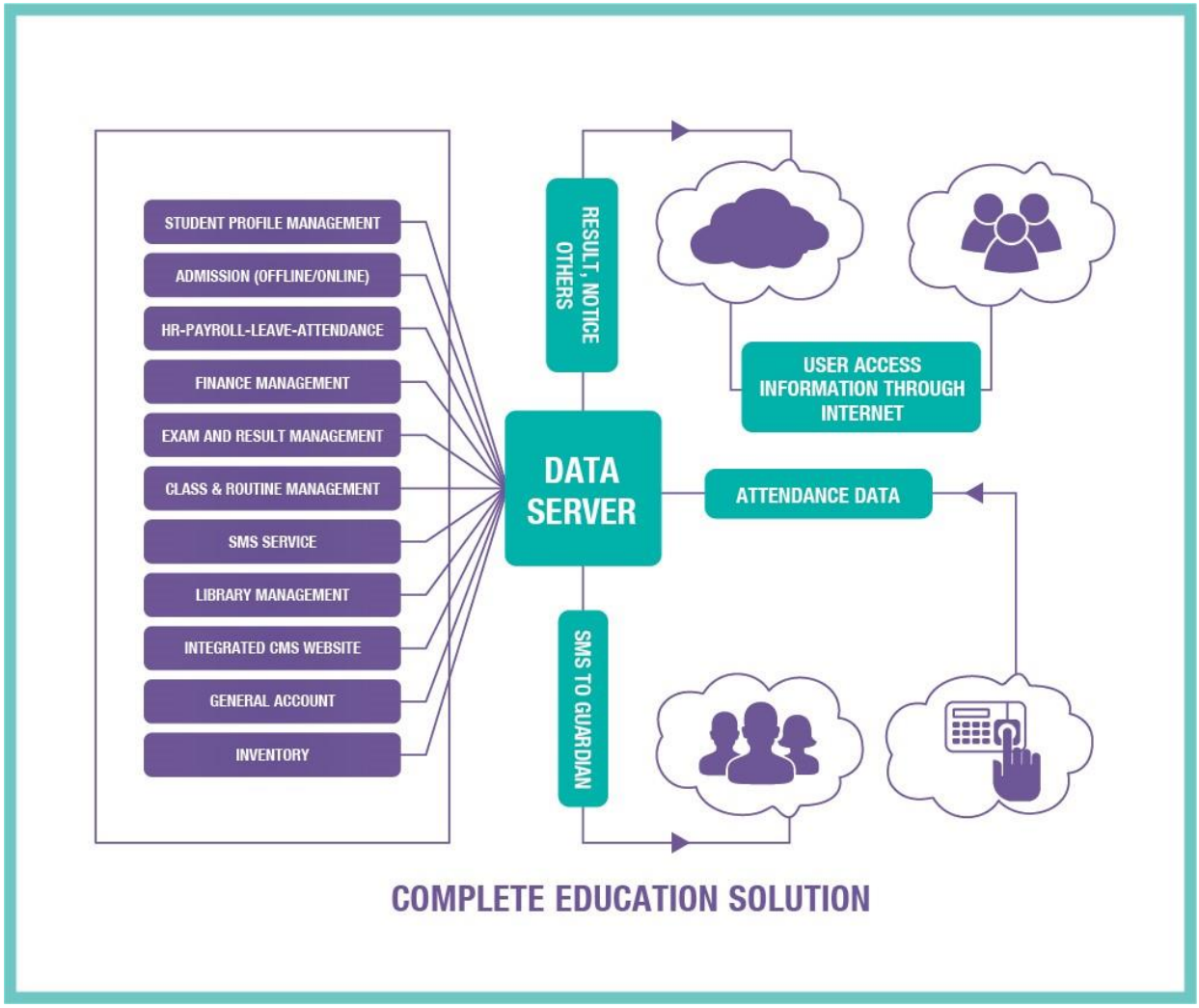

Рисунок 1.6 – Схема роботи ODEMS

Недоліком даної системи є те, що вона не підтримується іншими мовами окрім англійської, а також недоступність в нашій країні.

Також варто упом'янути ресурс для розкладу університету «КПІ» - rozklad.kpi.ua. Даний ресурс надає інформацію про заняття студентам певної групи у зручно агрегованому вигляді. Також користувачі можуть

отримати інформацію про безпосередньо розклад викладачів, шукаючи за іменем, або використовуючи посилання з вибраного уроку. Ще однією можливістю даної системи є отримання розкладу екзаменів. В якості розширення поточного сервісу було запущено rozklad.org.ua, який має більш зручний та інтуїтивно зрозумілий інтерфейс та має дещо розширений функціонал, наприклад надає змогу оцінювати викладачів. Даний сервіс не є повноцінним сервісом для підтримки навчальної діяльності, проте може бути використаний як допоміжний сервіс під час організації навчального процесу.

2.2 Основні принципи автоматизації навчальної діяльності

Провідну роль у системі управління університетом відіграє інформаційний модуль, який визначає результативність, правильність та ефективність як функцій управління, так і інших модулів керування навчальним закладом.

Для створення ефективного застосунку керування університетом необхідно мати чітке уявлення про мету, цілі та способи їх досягнення в цій системі, що зручно і цілком природно можна описати як послідовність конкретних процесів (функцій), формально представлених у вигляді бізнес-процесів, що забезпечують діяльність університету. Усі бізнес-процеси повинні бути реалізовані засобами програмно-апаратних комплексів, які, як нам відомо, функціонують на базі програмних модулів, що обробляють певні структури даних або інформаційні об’єкти.

Оглянувши та розібравши вище вказані наявні системи, я зміг виділити основні принципи в їх побудові та функціонуванні.

Визначаються такі основні принципи автоматизації навчальної діяльності:

– використання закритого сховища навчально-методичного забезпечення навчального процесу;
– розбиття компонентів системи на модулі;
– використання даних із загального сховища даних (інтегрована БД) з розмежуванням прав доступу на рівні користувачів і окремих додатків (окремих показників);
– використання загальних довідників;
– обмін інформацією між підсистемами на основі єдиного інформаційного середовища.

Саме за такими принципами проводиться побудова вище вказаних та подібних систем для підтримки навчальної діяльності навчальних закладів.

Також не менш важливим принципом в сучасному світі є принцип відкритості та публічності. Це зумовлено однією з основних ознак сучасного університету, а саме, наявністю відкритих для користування якісних електронних освітніх та наукових ресурсів.

Практика проектування інформаційних систем показала, що одним з найбільш ефективних методів проектування структур даних є документо-орієнтований підхід, оскільки діяльність будь-якої соціально-економічної системи прийнято відображати у відповідних документах.

2.3 Проблеми організації автоматизованої підтримки навчальної діяльності

Не зважаючи на стрімкий розвиток сфери автоматизованої підтримки навчальної діяльності в цілому, та інструментів, які з часом набувають все більш завершеного вигляду та мають широкий функціонал, останні мають певні недоліки, спричинені певними складнощами в навчальних процесах.

Якщо ж розглядати загальні проблеми організації навчального процесу, то потрібно, перш за все, визначити цілі та задачі такого процесу, сформулювати основні проблеми, що потребують розв’язку, виявити також основних учасників цього процесу, визначити їх ролі та методи взаємодії

між ними. Це перший етап аналізу методичної системи організації. На другому ж етапі потрібно виявити шляхи розв’язання основних проблем та запропонувати відповідні для удосконалення системи.

Однією з проблем, є проблема формування розкладу, оскільки в цьому питанні потрібно враховувати багато деталей між учасниками, а також розробити оптимальний алгоритм реалізації. Дана проблема є також не лише в системі, а й в навчальному закладі, тому що за все це відповідає певна група людей, а людям як відомо властиво допускати помилки.

Іншою проблемою може бути виникнення великого навантаження на базу даних. У такому випадку його зниження можна досягти за допомогою двох основних способів. Перший – це розбиття бази даних на окремі частини, які фізично зберігатимуться на окремих серверах. При цьому виникають дві основні проблеми: дотримання цілісності взаємопов’язаних даних, які знаходяться на різних серверах; рівномірний розподіл навантаження на кожен окремий сервер. Другий, більш прийнятний, спосіб – це використання кластерних технологій з автоматичним балансуванням навантаження. У цьому випадку кожен з двох і більше серверів баз даних містить повну копію всієї бази даних, спеціальний балансувач навантаження рівномірно розподіляє запити між усіма серверами, у разі зміни даних на одному із серверів система реплікації в режимі реального часу синхронізує всі сервери баз даних.

Також великою проблемою є залежність таких систем від інтернет підключення, оскільки без нього вони просто-на-просто не працюють.

2.4 Висновки

В даному розділі було досліджено принципи автоматизації навчальної діяльності кафедри, проаналізовано наявні засоби та інструменти в цьому питанні, найзручнішим з яких є, на мою думку, електронний кампус.

Підсумовано наявні можливості та властивості таких систем. Також

виділено основні проблеми при автоматизації навчальної діяльності.

З огляду на наявні проблеми було вирішено розробити застосунок, який відповідатиме основним критеріям:

– надійність і безпека;
– сумісність;
– зручність у використанні та адмініструванні;
– модульність;
– забезпеченість доступу.

Перейдемо до розгляду проектування системи.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ НАВЧАЛЬНОЇ ДІЯЛЬНОСТІ

Питання побудови системи взаємодії учасників такої системи полягає у тому, щоб визначити можливості, якими буде володіти така система, а також проблеми, які вона вирішуватиме.

При побудові такої системи потрібно звернути увагу на наступні складнощі:

– система повинна надавати змогу вносити інформаційні зміни;

– система має бути доступною для легкої інтеграції та оновлення.

З огляду на це розглянемо етапи проектування даної системи. Спроектуємо концептуальну модель бази даних та сформуємо архітектуру майбутнього застосунку, а також розглянемо деякі компоненти.

3.1 Моделювання роботи системи

Взаємодію користувача з системою зображено на рисунку 3.1.

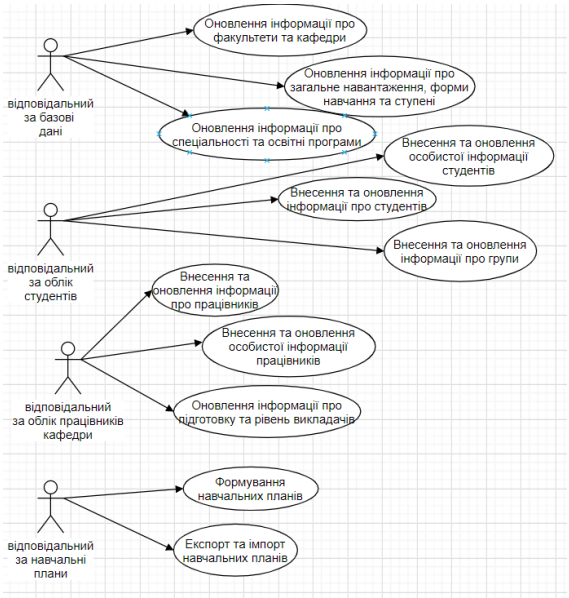

Рисунок 3.1 – Діаграма прецедентів системи

Оглянувши його, стає зрозуміло, що користувачі системи матимуть свою роль в майбутньому застосунку та відповідний функціонал, який є доступним для тієї ролі.

Розглянемо детальніше одну з ролей додатку, а саме роль працівника навчального відділу. Діаграму прецедентів зображено на рисунку 3.2.

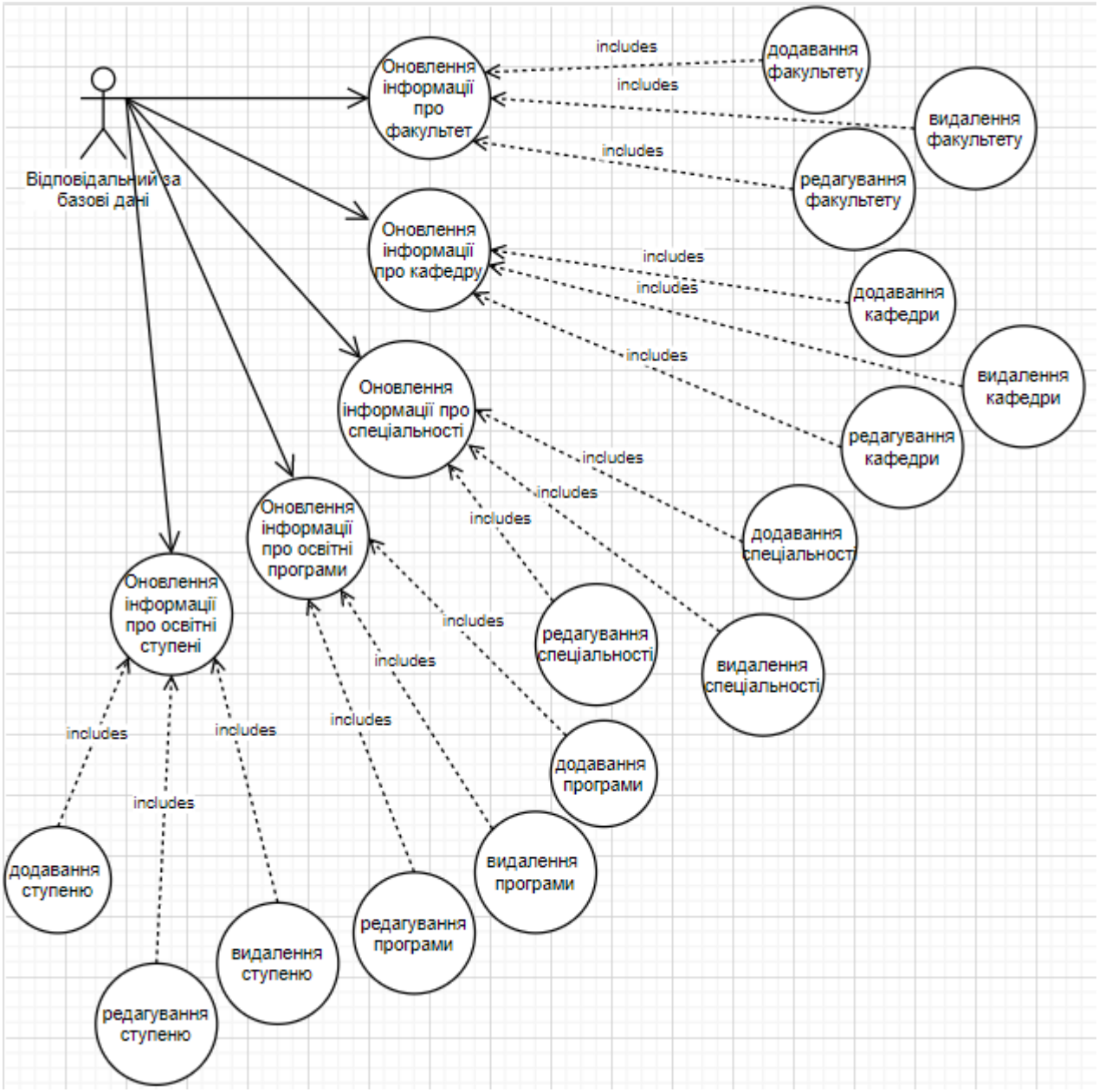

Рисунок 3.2 – Діаграма прецедентів

Система взаємодії між учасником автоматизації навчальної діяльності має наступні вимоги, зображені на концептуальній діаграмі прецедентів, де в якості актора виступає «Працівник навчального відділу».

Кожен з прецедентів діаграми відповідає за оновлення інформації

певних таблиць бази даних. Дана роль важлива оскільки, на внесених тут даних, будуть залежати деякі інші ролі, які для своєї роботи повинні використовувати ці дані.

Також для більшого розуміння як виконуватиметься робота з програмою, на рисунку 3.3 зображено діаграму діяльності, яка показує, як саме користувач взаємодіятиме з програмою та що від нього вимагається.

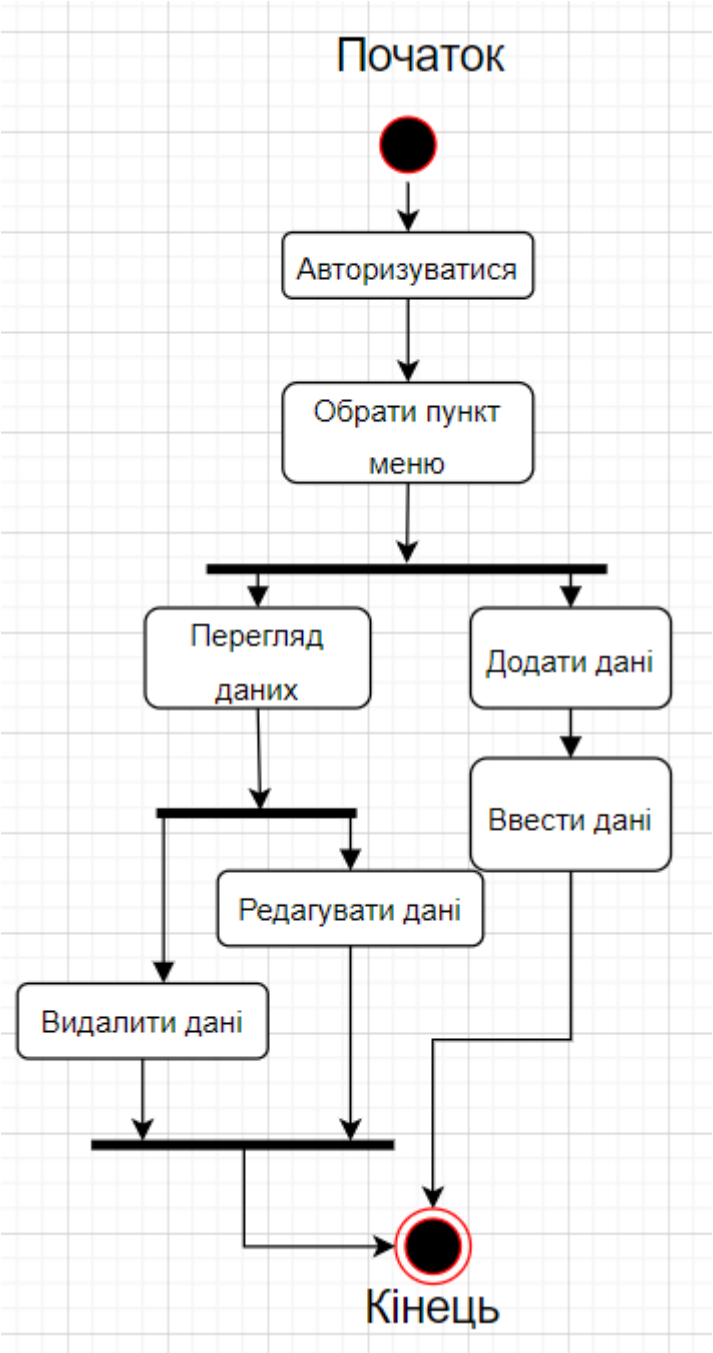

Рисунок 3.3 – Діаграма діяльності

3.2 Архітектура системи

Система являється сервісом, який надає змогу формувати базові дані у навчальному відділі, які в подальшому можуть використовуватися для створення та формування навчальних планів.

Орієнтуючись на вимоги до системи, отримаємо наступний вигляд архітектури системи, показаної на рисунку 3.4.

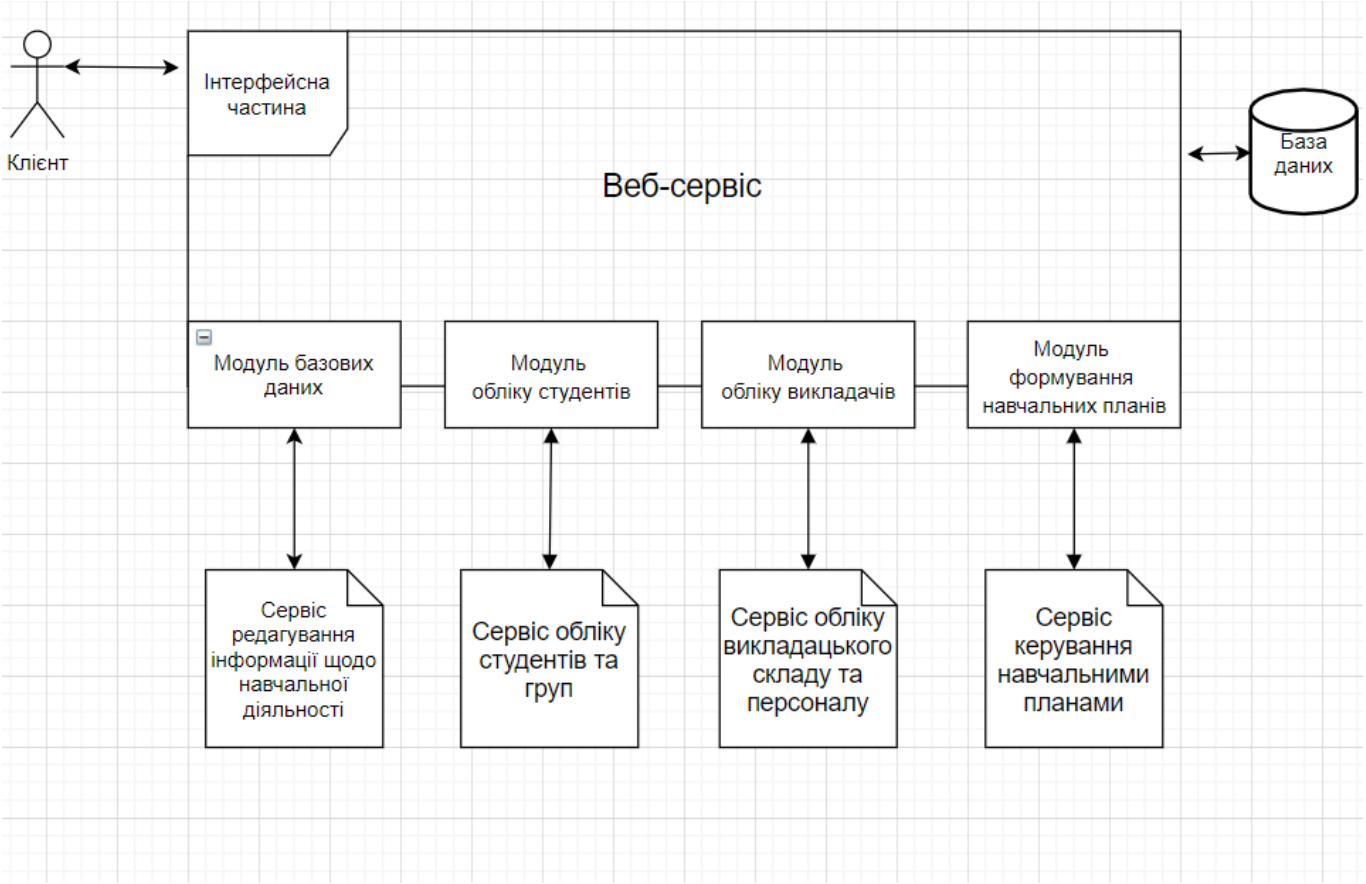

Рисунок 3.4 – Архітектура системи

Система являється веб-сервісом, який користувачі мають змогу використовувати з будь-яких зручних для них пристроїв, якщо вони мають можливість доступу до Інтернету, а також анігілює потребу у встановленні та налаштуванні системи.

Веб-сервіс містить у собі підмодуль, який відповідає за оновлення інформації у навчальному відділі, тобто додавання, редагування, видалення та перегляд записів у таблицях.

Також для більшого розуміння архітектури та того, як працюватиме система, було спроектовано діаграму розгортання, зображену на рисунку 3.5.

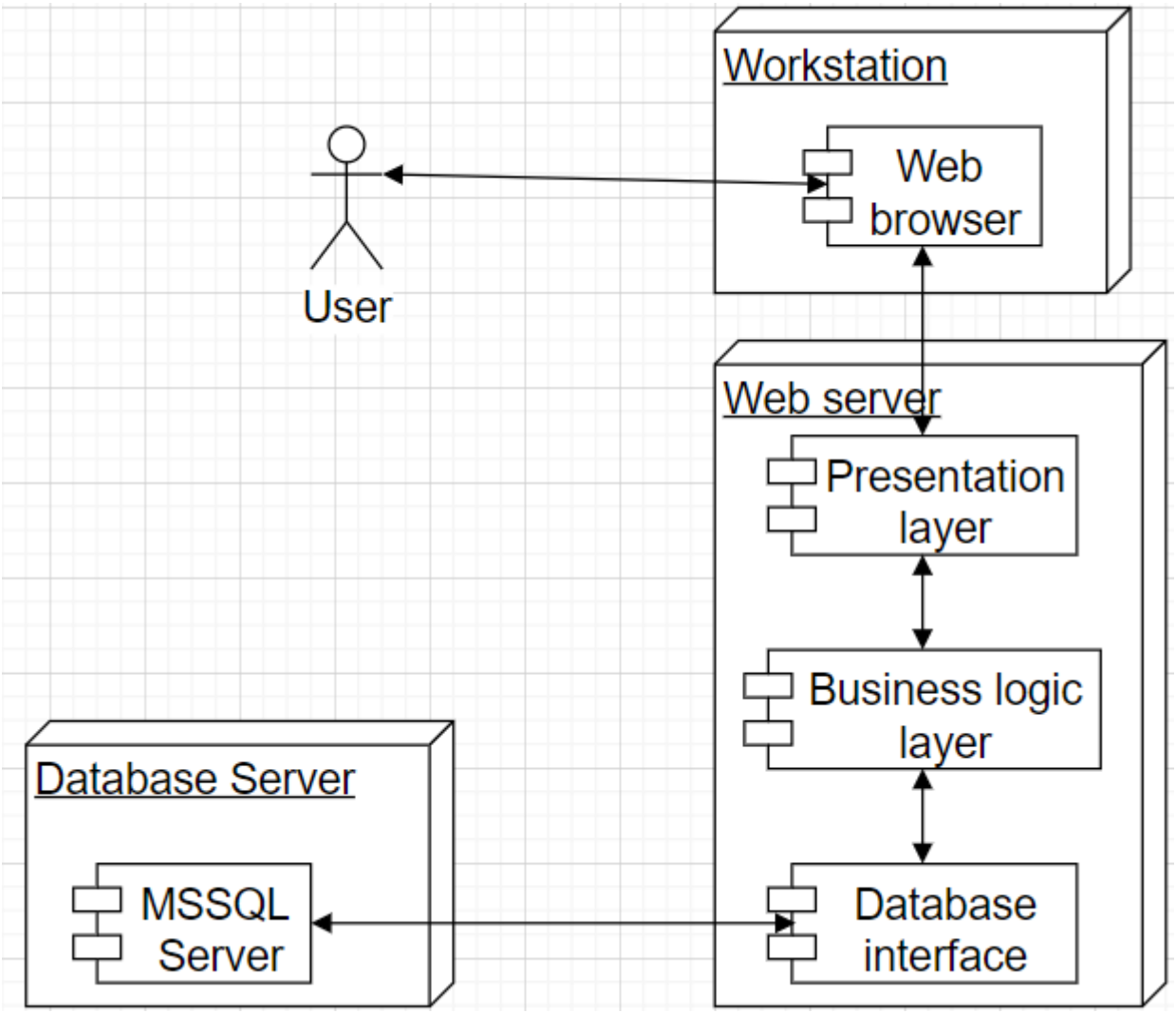

Рисунок 3.5 – Діаграма розгортання

3.3 Концептуальна модель бази даних

Першим етапом процесу проектування бази даних для інформаційних систем називають концептуальним проектуванням бази даних. Його основна мета це створення концептуальної моделі даних предметної області. Дуже важливо зробити хорошу модель бази даних, щоб при самій її реалізації не повертатися назад та не виправляти помилки.

Для нормальної організації та керуванням наявною інформацією програма повинна мати своє власне сховище у вигляді бази даних. На

рисунку 3.6 зображена загальна база даних, розроблена для повноцінної системи з різними модулями. Розглянемо таблиці, які використовуються у даному модулі (таблиці 3.1 – 3.7).

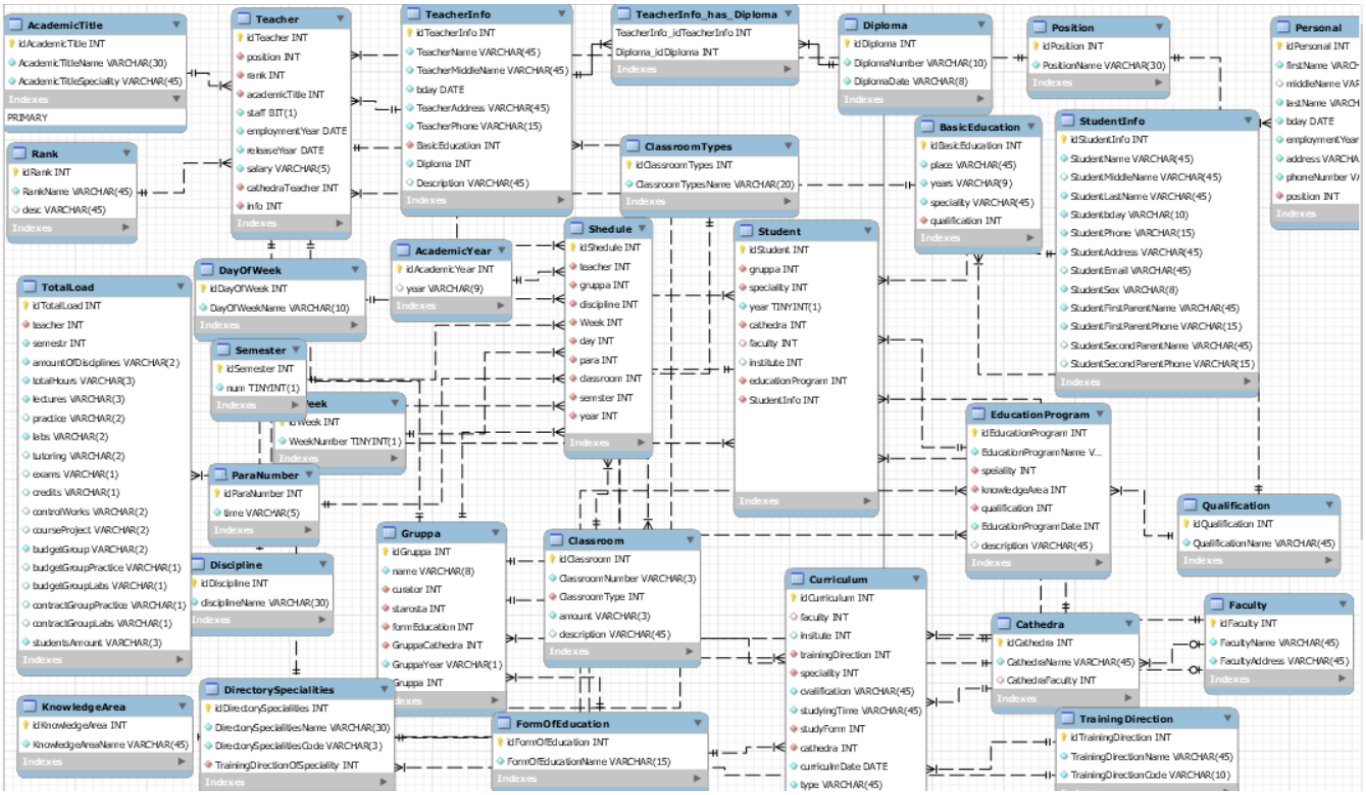

Рисунок 3.6 – Концептуальна модель бази даних

Таблиця «Факультети», зображена на в таблиці 3.1, відповідає за збереження мінімальної інформації про факультети. Вона складається з трьох полів, якими є унікальний ідентифікатор IdFaculty типу Integer для швидкого пошуку в таблиці, поля facultyName та facultyAddress зберігатимуть текстові дані, а саме назву факультету та адресу за якою він зареєстрований.

Таблиця 3.1 – Таблиця «Факультети»

№	Назва поля	Тип поля	Опис
1	IdFaculty	Int	Первинний ключ таблиці та ідентифікатор
2	facultyName	Varchar(25)	Назва факультету
3	facultyAddress	Varchar(25)	Адреса факультету

Таблиця «Кафедри» (Таблиця 3.2) містить інформацію кожної кафедри, а саме IdCathedra – зберігає ідентифікатор кафедри типу Int, cathedraName містить текстове поле, в якому буде знаходитися назва кафедри та ID_faculty, цілочисельне поле типу Integer, за допомогою якого таблиця пов’язуватиме кафедру та факультет, якому вона належить.

Таблиця 3.2 – Таблиця «Кафедри»

№	Назва поля	Тип поля	Опис
1	IdCathedra	Int	Первинний ключ та ідентифікатор
2	cathedraName	Varchar(25)	Назва кафедри
3	ID_faculty	Int	Зовнішній ключ, що пов’язує кафедри та факультети

Таблиця «Спеціальності» містить інформацію про спеціальності, за якими здійснюється навчальний процес. Вона містить текстові поля для specialityName та specialityCody, які відповідно зберігають назву та код спеціальності. Також в таблиці наявні два ключі цілочисельного типу, а саме IdSpeciality, потрібний для швидкого пошуку спеціальності, та ID_KnowledgeArea, який вказує до якого напряму відноситься кожна спеціальність. Дану таблицю зображено в таблиці 3.3.

Таблиця 3.3 – Таблиця «Перелік спеціальностей»

№	Назва поля	Тип поля	Опис
1	IdSpeciality	Int	Первинний ключ та ідентифікатор таблиці
2	specialityName	Varchar(30)	Назва спеціальності
3	specialityCode	Varchar(3)	Код спеціальності
4	ID_KnowledgeArea	Int	Зовнішній ключ для пов’язування з напрямом підготовки

Таблиця «Напрям підготовки» (таблиця 3.4) зберігає інформацію про підготовчі напрями – його назву (areaName – текстове поле), код (areaCode – текстове поле) та його назву (areaName – текстове поле), код (areaCode – текстове поле) та ідентифікатор (IdArea – цілочисельне поле).

Таблиця 3.4 – Таблиця «Напрями підготовки / галузі знань»

№	Назва поля	Тип поля	Опис
1	IdArea	Int	Первинний ключ та ідентифікатор таблиці
2	areaName	Varchar(25)	Назва напряму підготовки
3	areaCode	Varchar(3)	Код напряму підготовки

Таблиця «Освітні програми» (таблиця 3.5) містить дані про наявні освітні програми. Загалом таблиця містить поля:

- IdProgram – Int – дане поле є первинним ключем таблиці;
- programName – Varchar(25) – назва освітньої програми;
- IdSpeciality – Int – це поле є вторинним ключем, який визначає спеціальність програми;
- IdKnowledgeArea – Int – вторинний ключ, тобто посилання на відповідний напрям підготовки у таблиці 3.4;
- IdQualification – Int – вторинний ключ, який визначає кваліфікацію програми;
- IdEducationalDegree – Int – це поле є вторинним ключем, який визначає освітній ступінь програми;
- AccessDate – Date – поле вказує на дату прийняття освітньої програми;
- Note – Varchar(25) – примітка для додаткової інформації про програму.

Таблиця 3.5 – Таблиця «Освітня програма»

№	Назва поля	Тип поля	Опис
1	IdProgram	Int	Первинний ключ та ідентифікатор таблиці
2	programName	Varchar(25)	Назва освітньої програми
3	IdSpeciality	Int	Зовнішній ключ для пов’язування з спеціальністю
4	IdKnowledgeArea	Int	Зовнішній ключ для пов’язування з напрямом підготовки
5	IdQualification	Int	Зовнішній ключ для

			пов’язування з кваліфікацією
6	IdEducationalDegree	Int	Зовнішній ключ для пов’язування з освітнім рівнем
7	AccessDate	Date	Дата затвердження програми
8	Note	Varchar(45)	Примітка для додаткової інформації

Таблиця «Освітній ступінь» відповідає за збереження даних про назву ступеня, а також унікальний ідентифікатор. Вона зображена в таблиці 3.6.

Таблиця 3.6 – Таблиця «Освітні ступені»

№	Назва поля	Тип поля	Опис
1	IdDegree	Int	Первинний ключ та ідентифікатор таблиці
2	degreeName	Varchar(25)	Назва рівня

Таблиця «Загальне навантаження» містить інформацію про навантаження викладача у певному семестрі. Також дана таблиця зберігає кількість дисциплін, загальну кількість годин, кількість лекцій, практик та лабораторних занять, курсові проекти, групи та кількість занять в них, а також загальну кількість студентів. Більш детально її можна оглянути в таблиці 2.7 та описі до неї.

Таблиця 2.7 – Таблиця «Загальне навантаження»

№	Назва поля	Тип поля	Опис
1	IdLoad	Int	Первинний ключ та ідентифікатор таблиці
2	IdTeacher	Int	Зовнішній ключ для пов’язування з викладачем
3	IdSemestr	Int	Зовнішній ключ для пов’язування з семестрами
4	AmountOfDiscipline	Varchar(1)	Кількість дисциплін, яку викладач веде
5	AmountOfDisciplineInSemestr	Varchar(3)	Кількість дисциплін у семестрі
6	Lectures	Varchar(3)	Загальна кількість лекцій

7	Practice	Varchar(2)	Загальна кількість практичних занять
8	Labs	Varchar(2)	Загальна кількість лабораторних занять
9	Tutoring	Varchar(2)	Загальна кількість індивідуальних занять
10	Exam	Varchar(1)	Загальна кількість екзаменів
11	Credit	Varchar(1)	Загальна кількість заліків
12	ControlWorks	Varchar(2)	Загальна кількість контрольних робіт
13	CourseWorks	Varchar(2)	Загальна кількість курсових робіт
14	BudgetGroups	Varchar(2)	Загальна кількість бюджетних груп
15	BudgetGroupsPractice	Varchar(3)	Загальна кількість практичних занять в бюджетних групах
16	BudgetGroupsLabs	Varchar(3)	Загальна кількість лабораторних занять в бюджетних групах
17	ContractGroups	Varchar(3)	Загальна кількість контрактних груп
18	ContractGroupsPractice	Varchar(3)	Загальна кількість практичних занять в контрактних групах
19	ContractGroupsLabs	Varchar(3)	Загальна кількість лабораторних занять в контрактних групах
20	AmountOfStudents	Varchar(3)	Загальна кількість студентів

Таблиця «Загальне навантаження» (таблиця 2.7) містить дані про загальне навантаження викладачів. Загалом таблиця містить поля:

- IdLoad– Int – дане поле є первинним ключем таблиці;
- IdTeacher – Int – зовнішній ключ, який вказує якого викладача стосуються дані в таблиці;
- IdSemestr – Int – це поле є вторинним ключем, яке визначає номер семестру;
- AmountOfDiscipline – Varchar(1) –поле вказує на загальну кількість дисциплін, які проводить викладач;

- AmountOfDisciplineInSemestr – Varchar(3) – зберігає кількість дисциплін в семестрі;
- Lectures – Varchar(3) – містить загальну кількість лекцій, які читає викладач;
- Practice – Varchar(2) – поле вказує на загальну кількість практик у викладача;
- Labs – Varchar(2) – містить загальну кількість лабораторних робіт, які проводить викладач;
- Tutoring – Varchar(2) – кількість індивідуальних занятть;
- Exam – Varchar(1) – загальна кількість екзаменів в кінці семестру, які викладач проводить;
- Credit – Varchar(1) – загальна кількість екзаменів в кінці семестру, які викладач проводить;
- ControlWorks – Varchar(2) – містить загальну кількість контрольних робіт;
- CourseWorks – Varchar(2) – поле вказує на загальну кількість курсових робіт у викладача;
- BudgetGroups – Varchar(2) – кількість бюджетних груп, в яких викладач проводить свою діяльність;
- BudgetGroupsPractice – Varchar(3) – вказує на кількість практик у бюджетних групах;
- BudgetGroupsLabs – Varchar(3) – вказує на кількість лабораторних робіт в таких групах;
- ContractGroups – Varchar(3) – поле вказує кількість контрактних груп;
- ContractGroupsPractice – Varchar(3) – вказує на кількість практик у контрактних групах;
- ContractGroupsLabs – Varchar(3) – вказує на кількість лабораторних робіт в таких групах;
- AmountOfStudents – Varchar(3) – містить загальну кількість студентів, яких викладач навчає.

Нижче на рисунку 3.7 зображено частину концептуальної моделі бази даних, що використовуватиметься для цієї ролі та вище описаних таблиць.

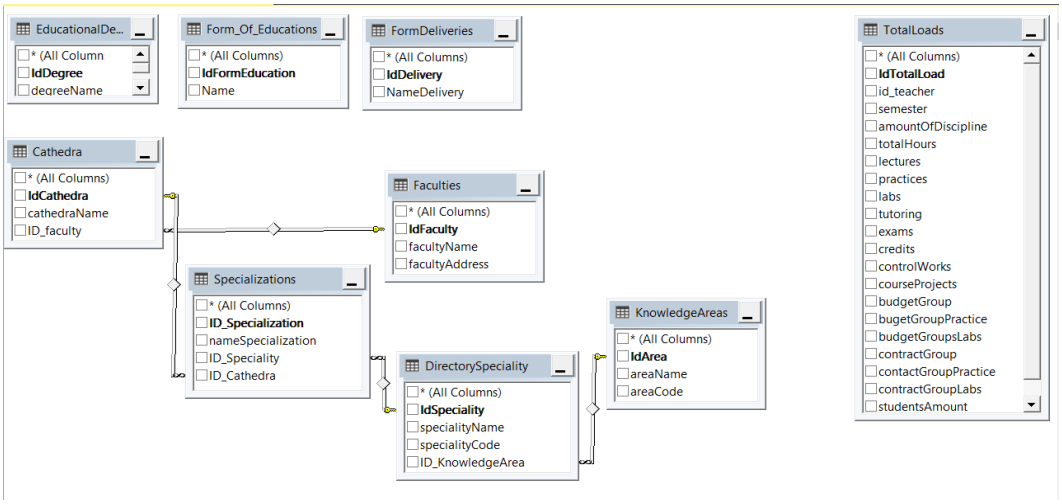

Рисунок 3.7 – Частина бази даних

3.4 Висновки до розділу 3

В даному розділі було розглянуто етапи проектування даної системи. Також було розглянуто складнощі, які виникли перед побудовою сервісу та загальний опис майбутньої системи. Також було побудовано вимоги до майбутнього застосунку та зображено у вигляді діаграми прецедентів, кожен з якої був розглянутий окремо.

Спираючись на вимоги, які були сформовані, було спроектовано архітектуру додатку та розглянуто наявні в ньому компоненти.

Також спроектовано концептуальну модель бази даних для даної програми та розглянуто її моделі для відповідного модулю системи.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ПІДТРИМКИ НАВЧАЛЬНОЇ ДІЯЛЬНОСТІ КАФЕДРИ

Даний додаток реалізує систему формування базових даних навчального відділу, які в подальшому будуть використовуватися різними модулями. Система має модульну структуру та використовує сервер бази даних, що забезпечує механізми збереження цілісності даних та легку адаптацію змін.

4.1 Вибір засобів розробки

Спроектована система буде являтися веб-сервісом, який складається з серверної та клієнтської частин. Вибір засобів розробки є важливим етапом перед початком розробки системи, тому потрібно зробити гарний вибір, щоб в подальшому зручно було додавати та оновлювати компоненти системи. Для розробки серверної частини існує чимало різних мов програмування, найбільш популярними з яких є PHP, C#, Java, Node.js, Python. Оглянувши всі переваги кожної мови та взявши до уваги спроектовані раніше концептуальну модель бази даних та архітектуру системи, для розробки серверної частини було обрано мову C# [28], оскільки він легше взаємодіє з кодом програм, написаних на інших мовах, C# більше підходить для об’ємних застосунків та пропонує всі необхідні розробнику інструменти найвищої якості.

C# - на сьогоднішній день являється однією з найпотужніших, швидко розвиваючих та затребуваних мов програмування в IT-сфері. Вона є сучасною досить популярною об’єктно орієнтованою мовою, але підтримує також компонентно-орієнтоване програмування, відома завдяки створенню корпоративного програмного забезпечення, фінансових

проектів, мобільних додатків, а також хмарних сервісів. Не варто забучвати про написання ігор на Unity за допомогою цієї мови.

Для розробки серверної частини C# містить фреймворки Entity Framework, в подальшому EF, та ASP.NET на платформах .Net Framework й .Net Core. Оскільки однією з вимог до майбутнього додатку було крос-платформенність, то використовуватиметься .Net Core, а не .Net Framework, який призначений для розробки тільки для операційної системи Windows.

Для розробки клієнтської частини існує також багато різних інструментів, які допоможуть зробити інтерфейс системи приємним та зручним для користувача. Тут варто виділити JS-фреймворки, такі як Angular, React та Vue.js. Це найбільш популярні серед них, але з них тільки один Angular співпрацює з нашою основною мовою серверної частини. Звісно можна використовувати й інші, але Angular та C# краще взаємодіють та простіше, але він призначений для великих проектів, тому його не варто використовувати в нашій системі. Не варто забувати також про фреймворки для клієнтської частини, які нам пропонує компанія Microsoft, адже мова C# це теж їх розробка, так само як найбільш потужне середовище для розробки на цій мові під назвою Visual Studio. Серед запропонованих фрейморків від Microsoft можна зустріти синтаксис Razor, SignalR та синтаксис Blazor. Оскільки серед цих трьох фреймворків лише тільки Razor призначений для нашого типу застосунку, то ми будемо використовувати його.

Отже, після довгого вибору засобів для розробки системи, було обрано мову програмування для серверної частини C#, а точніше його фреймворк для розробки веб-застосунків ASP.NET Core, для керуванням базою даних та здійснення операцій з нею використовуватиметься EF Core, а саме один з його підходів Code first, і нарешті для клієнтської частини було обрано синтаксис Razor.

4.1.1 Entity Framework Core

Entity Framework Core (EF Core) є об’єктно-орієнтованою, легкою,

переносною розширюваною технологією від компанії Microsoft для доступу до даних, а також створення та керування даними [17]. EF Core являється ORM-інструментом (object-relational mapping – відображення даних на реальні об'єкти), тобто EF Core дає змогу працювати з базами даних, але він представляє собою вищий рівень абстракції: EF Core дозволяє абстрагуватися від самої бази даних та її таблиць і працювати з даними незалежно від типу сховища. Якщо ж на фізичному рівні бази даних ми оперуємо таблицями, індексами, первинними та зовнішніми ключами, то на концептуальному рівні, пропонованим даним фреймворком, ми оперуємо самими об'єктами.

Entity Framework Core підтримує багато різних систем керування базами даних, в тому числі найпопулярніші MySQL, MS SSMS та PostgreSQL. Таким чином, ми можемо через EF Core працювати з будь-якою СУБД, якщо для неї є дійсний провайдер.

Також варто відмітити, що EF Core пропонує універсальний API для роботи з даними, так званий FluentAPI.

Центральною концепцією Entity Framework є поняття сутності або entity. Сутність визначає набір даних, які пов'язані з визначеним об'єктом. Тому дана технологія пропонує роботу не з таблицями, а з об'єктами та їх колекціями. Будь-яка сутність має ряд властивостей, які будуть відрізняти цей об'єкт від інших та будуть унікально визначати цей об'єкт. Відмінною властивістю EF Core, як ORM технології, являється використання LINQ запитів для вибірки даних із бази даних. За допомогою LINQ ми можемо створювати різні запити на вибірку об'єктів, в тому числі пов'язаних різними асоціативними зв'язками. А EF Core при виконанні запитів транслює вираз LINQ у запит, зрозумілий для конкретної СУБД.

Також EF Core має три підходи для роботи, до них належать:

– code first, використовується, коли вже є існуючий застосунок, який зберігає модель даних;
– database first, дозволяє писати додатки для існуючих баз даних, формує сутності з існуючих таблиць та створює контекст даних;

– model first, використовується, коли розробники не хочуть використовувати інструменти СУБД для створення і керування базою даних, а також не хочуть вручну налаштовувати класи моделі EDM.

4.1.2 Платформа .NET Core

Платформа .NET Core – це технологія, яка підтримує створення і виконання нового покоління додатків і веб-служб XML [12]. При розробці платформи .NET Core були поставлені наступні цілі:

– кросплатформенність. Підтримка різних операційних систем, таких як Linux та macOS.

– Відкритий вихідний код. Платформа .Net Core базується на відкритому коді та розповсюджується по ліцензіях MIT та Apache 2.

– Сучасні технології. В ній реалізовані сучасні парадигми, такі як асинхронне програмування та інші.

– Продуктивність. Забезпечує високу продуктивність за рахунок таких можливостей, як влаштовані програмні компоненти, багаторівнева компіляція та Span<T>.

– Узгодження між середовищами. Однакове виконання коду в різних операційних системах та архітектурах x64, x86 та ARM.

– Програми командного рядка. Зручні засоби командного рядка для локальної та неперервної інтеграції.

– Гнучка розробка. .Net Core можна включити в додаток або встановити паралельно. Можливість використання з контейнерами Docker.

Саме за допомогою цієї платформи, написаний мною код конвертується в машинні операції, що дозволяє машині (ноутбуку) виконувати безперервну роботу, розробленої мною програми.

4.1.3 Фреймворк ASP.NET Core

Платформа ASP.NET Core представляє технологію від компанії Microsoft, призначену для різного роду веб-додатків: від невеликих веб-сайтів до великих веб-порталів та веб-сервісів. Вона є повністю opensource-фреймворком. Всі вихідні файли можна розглянути на Github.

За допомогою ASP.NET Core можна створювати крос-платформенні додатки. Завдяки модульності фреймворку всі необхідні компоненти вед-застосунку можуть завантажуватися як окремі модулі через пакетний менеджер Nuget. ASP.NET Core включає в себе фреймворк MVC, який об'єднує функціональність MVC, Web API та Web Pages. Крім вище згаданих технологій в одну модель в нього також добавлено ряд додаткових функцій. Одною з таких функцій є тег-хелпери(tag helper), які дозволяють більш органічно поєднувати синтаксис HTML з кодом C#.

4.1.4 Середовище розробки Visual Studio 2019

Середовище Visual Studio 2019 – це інтегроване середовище розробки програмного забезпечення від компанії Microsoft [11]. За допомогою Visual Studio можна створювати додатки для Windows, iOS, Android та інших платформ. У Visual Studio включені інструменти не тільки для створення desktop додатків, але і web, мобільні і хмарні інструменти розробки. Є можливість написання коду на таких мовах як: C ++, C #, Visual Basic, F #, JavaScript, Python, TypeScript. Крім усього цього вона включає в себе конструктори, редактори, відладчики, а також величезну кількість розширень для різних областей застосування - від PHP до ігор.

На рисунку 4.1 зображено інтерфейс Visual Studio 2019.

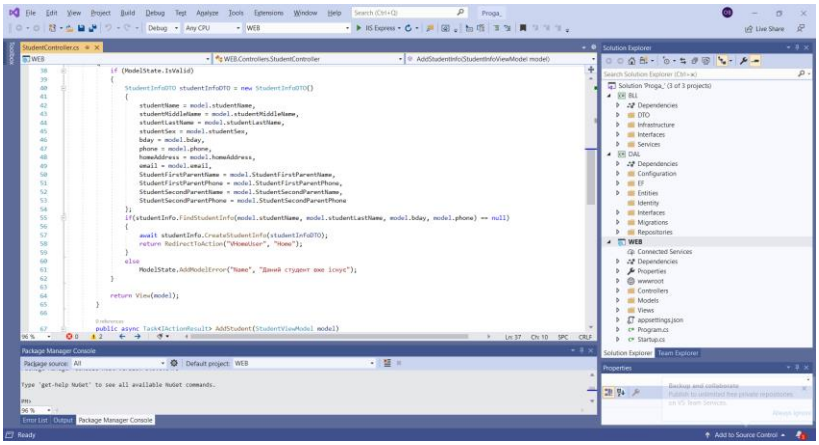

Рисунок 4.1 – інтерфейс Visual Studio 2019

Середовище Visual Studio 2019 року включає в себе наступні нові можливості і поновлення:

– випускається в трьох редакціях (раніше було чотири);

– крос-платформна підтримка мобільних пристроїв (Android, IOS, і Windows);

– покращення в C ++;

– зміни в налагодженні і діагностиці;

– платформа .NET Framework 4.8;

– платформа .NET Core 3.1;

– покращена інтеграція Visual Studio і GitHub;

– і безліч інших.

4.1.5 HyperText Markup Language(HTML)

Для написання будь-якого шаблону представлення використовуватиметься HTML. Ця мова розшифровується як HyperText Markup Language, що в перекладі означає – мова гіпертекстової розмітки. Вона існує з 1992 року[27].

Документи цього формату представляють звичайні текстові файли за винятком того, що деякі символи інтерпретуються як розмітка. Опис сторінки за допомогою цієї мови є набором інструкцій. Дані інструкції пишуться за допомогою так званих тегів (tag), які веб-браузер

інтерпретує та відображає користувачеві. Відповідно після таких дій HTML-файл стає Web-документом.

Дана мова використовується майже всіма веб-ресурсами, адже клієнтська частина пишеться завдяки їй.

4.1.6 Cascading Style Sheets(CSS)

Як і HTML, CSS насправді не є мовою програмування, а каскадними таблицями стилів. Це означає, що вони дозволяють застосовувати стилі вибірково до елементів та блоків у HTML-файлах. Наприклад, щоб вибрати всі елементи меню на сторінці HTML і зробити текст у них жовтим кольором.

CSS дозволяє створювати правила, які вказують, як елемент повинен з'явитися та як буде виглядати. Наприклад, можна вказати, що колір фону сторінки буде темно-синім, всі параграфи повинні використовувати шрифт Times New Roman або всі рівні першого заголовки повинні бути червоні та виділені курсивом [30].

CSS дозволяє виносити описи стилів, що повторюються, одних і тих самих елементів у окремий файл. Завдяки цьому відбувається значне "розвантаження" документів HTML. HTML-код стає легким, зручним для читання та редагування.

4.1.7 СУБД SQL Server Management Studio

SQL Server Management Studio (SSMS) – це інтегроване середовище для керування будь-якою інфраструктурою SQL. Служба SSMS включає в собі широку групу графічних інструментів з рядом чудових редакторів сценаріїв для забезпечення доступу до служби SQL Server для розробників та адміністраторів всіх професійних рівнів.

Дане середовище потрібне для проектування концептуальної моделі бази даних, опису майбутніх компонентів бази даних, а також це середовище виступатиме у ролі локального серверу, до якого система підключатиметься та буде зберігати дані, з якими працюватиме. Локальний сервер це інструмент веб-розробника для створення та налагодження різних скриптів та веб-застосунків. Тобто він є емулятором реального серверу хостинг провайдера, який розміщений на нашому комп'ютері. Головне вікно середовища зображено на рисунку 4.2.

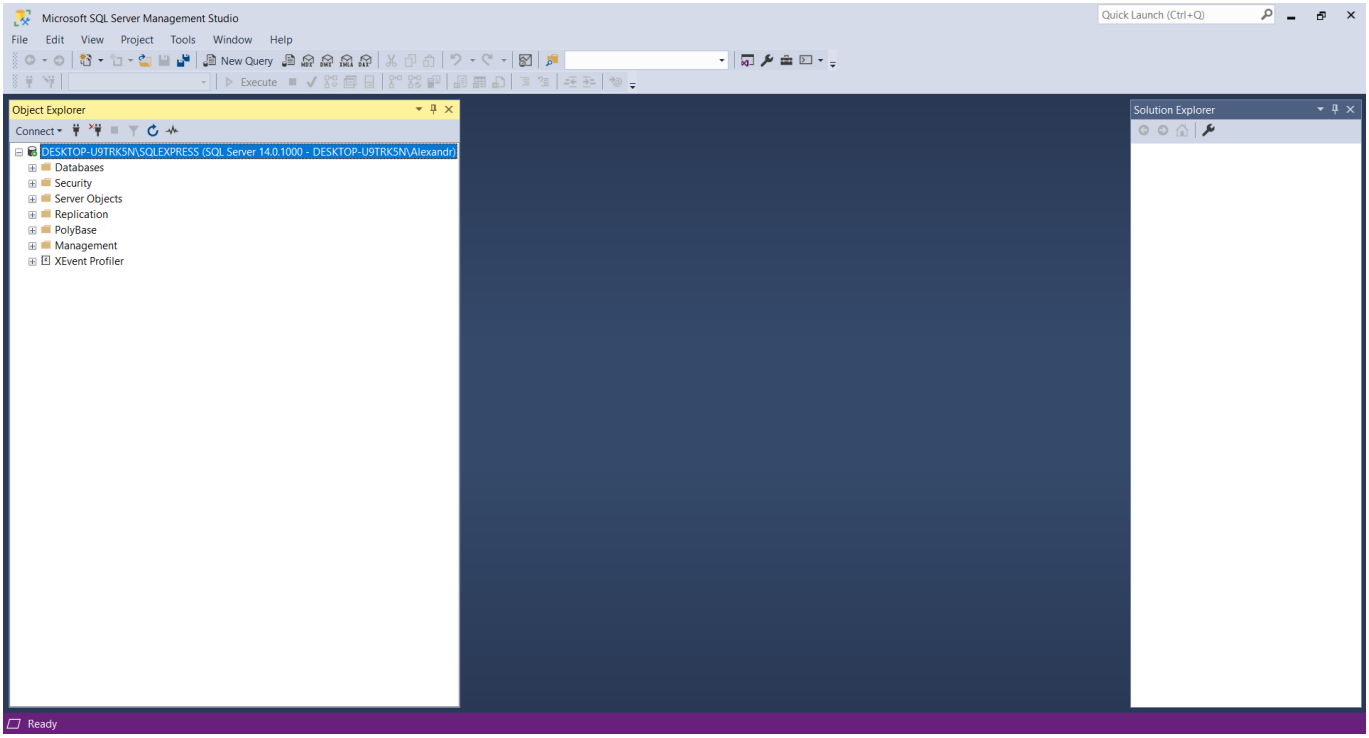

Рисунок 4.2 – Головне вікно середовища SSMS

4.2 Вибір патерна проектування

Після вибору засобів розробки, важливим етапом є вибір патерна проектування, за допомогою якого будуватиметься майбутня система. C# підтримує всі три види патернів, а саме:

– породжуючі патерни;
– патерни поведінки;
– структурні патерни.

Але мало з цих патернів підійдуть для конкретної задачі, оскільки призначені для трохи інших типів задач, деякі занадто складні, щоб їх використовувати.

В додатках ASP.NET часто використовується патерн «репозиторій» для інкапсуляції логіки роботи з джерелом даних. І часто ми оперуємо великою кількістю сутностей і моделей, для керування якими створюється також багато класів-репозиторіїв. Патерн Unit of Work дозволяє спростити роботу з різними репозиторіями і дає змогу бути впевненим, що всі репозиторії будуть використовувати один і той самий контекст даних. Також цей патерн більше всього підійде для даної задачі, оскільки ми оперуватимемо великою кількістю даних та потребуємо тільки один контекст даних. Принцип роботи цього патерну показано на рисунку 4.3.

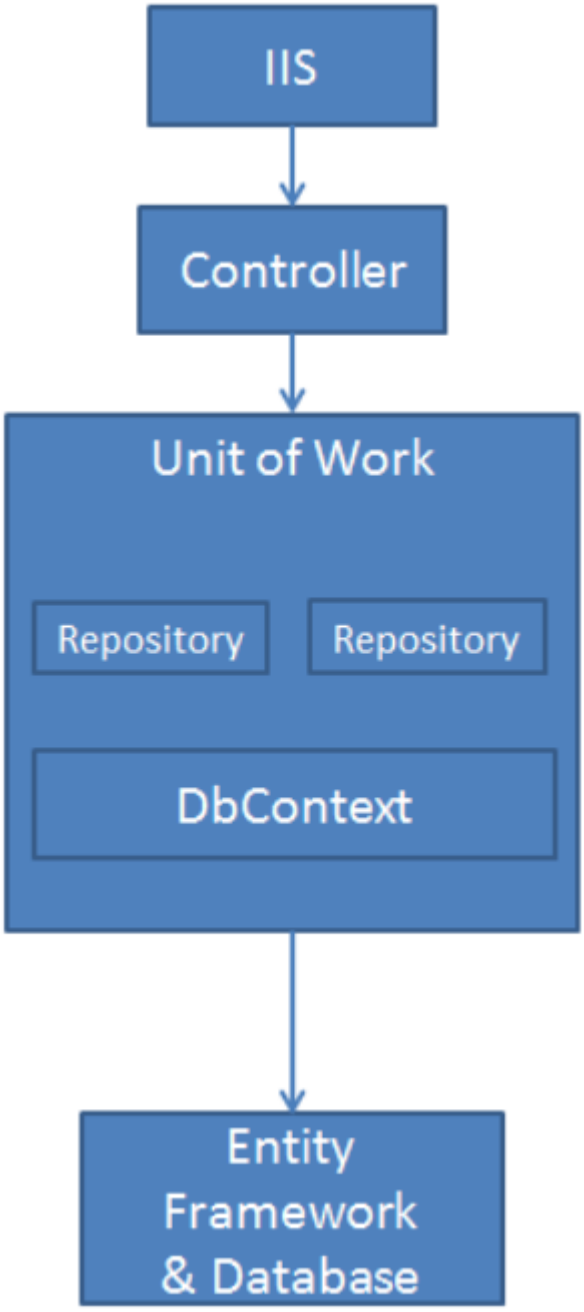

Рисунок 4.3 – Патерн проектування Unit of Work

Частину коду з репозиторію UnitOfWork показано на рисунку 4.4.

```
public async Task SaveAsync()
{
    await context.SaveChangesAsync();
}


public UnitOfWork(DbContextOptions connectionString)
{
    context = new ApplicationContext(connectionString);
}


public IRepositoryMain<User, string> RUsers
{
    get
    {
        if (userRepository == null)
            userRepository = new UserRepository(context);
        return userRepository;
    }
}
```

Рисунок 4.4 – Частина класу UnitOfWork

4.3 Загальний опис компонентів

Додаток агрегує у собі дві частини – серверну та клієнтську. Серверна частина складається з двох проектів, які зображено на рисунку 4.5.

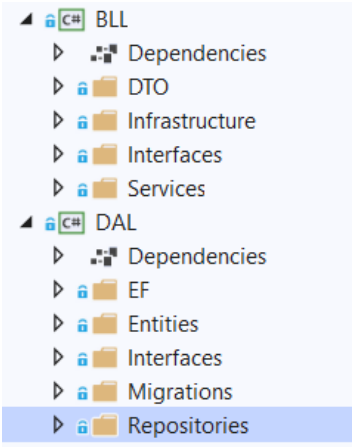

Рисунок 4.5 – Папки серверної частини

У проекті DAL, що розшифровується як data access layer, що в перекладі означає – рівень доступу даних. Цей проект слугує для створення бази даних, операцій з нею та її оновленням. Даний проект містить наступні папки:

– Dependencies, містить залежності та фреймворки, які використовуються в даному рівні;
– EF, зберігає контекст даних, потрібний для створення та роботи з базою даних;
– Entities, містить класи, з яких сформувалися таблиці в нашій БД. Перелік всіх сутностей зображено на рисунку 4.6.
– Interfaces, містить інтерфейси, які використовуються патерном проектування;
– Migrations, складається з класів, які містять код створення бази даних та її оновлення. Приклад створення таблиці показано на рисунку 4.7.
– Repositories, містить класи репозиторіїв до кожної таблиці, за допомогою них проектуються операції з базою даних.
–

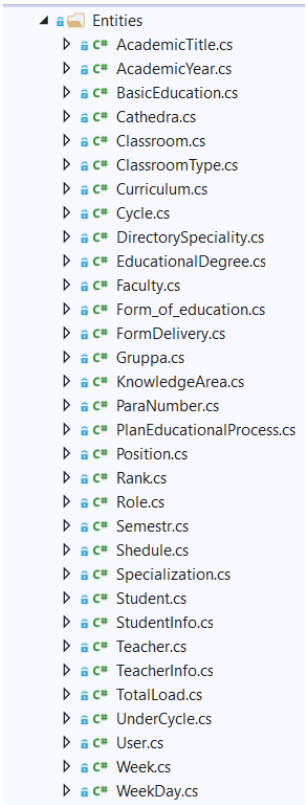

Рисунок 4.6 – Перелік сутностей

```
migrationBuilder.CreateTable(
    name: "AcademicTitles",
    columns: table => new
    {
        IdAcademicTitle = table.Column<int>(nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
        academicTitleName = table.Column<string>(nullable: true)
    },
    constraints: table =>
    {
        table.PrimaryKey("PK_AcademicTitles", x => x.IdAcademicTitle);
    });
```

Рисунок 4.7 – Приклад частини міграції

Проект BLL (business logic layer – рівень бізнес логіки), цей рівень потрібний для того, щоб пов’язати рівень представлення та рівень доступу даних. Він містить такі папки:

– Dependencies, містить залежності та фреймворки, які використовуються в даному рівні, а також підключений рівень доступу даних;чё
– DTO, ця папка містить так звані проміжкові моделі, для передачі даних з рівня представлення, тобто даних введених користувачем, на рівень доступу даних для додавання інформації та навпаки, з рівня доступу даних на рівень представлень для відображення інформації, яка зберігається;
– Infrastructure, містить класи, які використовуються для вказання деталей операцій та оброблення помилок, які можуть виникати;
– Interfaces, папка містить інтерфейси, за якими будуються сервіси для роботи системи;
– Services, самі сервіси, які імплементують інтерфейси, виконують обробку даних та передають їх.

Проект WEB реалізує presentation layer – рівень представлень, є найбільшим, оскільки самі представлення є більш об’ємними та містять додаткові складові для потрібні для повного функціонування. Він містить:

- Dependencies, так само як і минулі два рівні містить підключені фреймворки, залежності, рівень бізнес логіки для співпраці з ним;
- Properties – містить один єдиний файл launchSettings.json, який містить налаштування запуску застосунку;
- wwwroot містить папки зі стилями та скриптами, які використовуються для надання сторінкам вигляду;
- Controllers містить контролери керування та переходу між сторінками;
- Models містить в собі моделі, які використовуються у представленнях;
- Views це самі представлення, з якими працює користувач;
- appsettings.json містить шлях підключення до сервера;
- Program.cs, саме з цього класу починається запуск програми;
- Startup.cs є вхідною точкою в програму ASP.NET Core.

Останній проект реалізований за допомогою архітектури MVC (Model-View-Controller), що означає модель-представлення-контролер. Це один з найвідоміших підходів у фреймворці ASP.NET. Його структуру показано на рисунку 4.8.

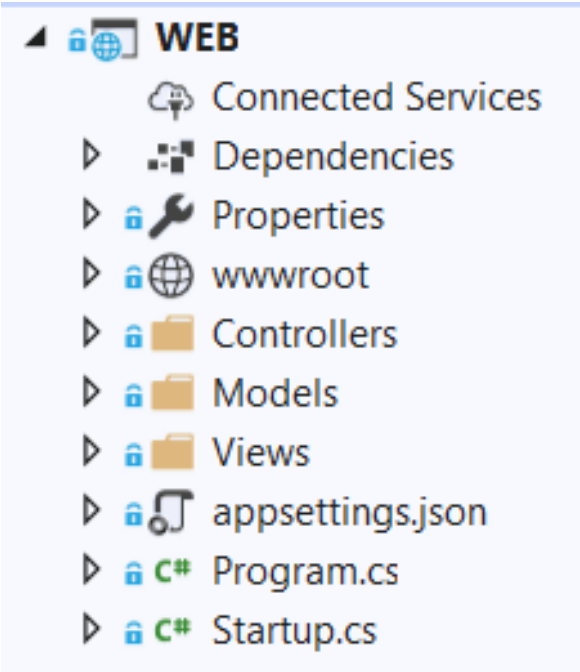

Рисунок 4.8 – Директорія клієнтської частини

Також загальну директорію системи можна побачити на рисунку 4.9 для більш зрозумілої структури програми.

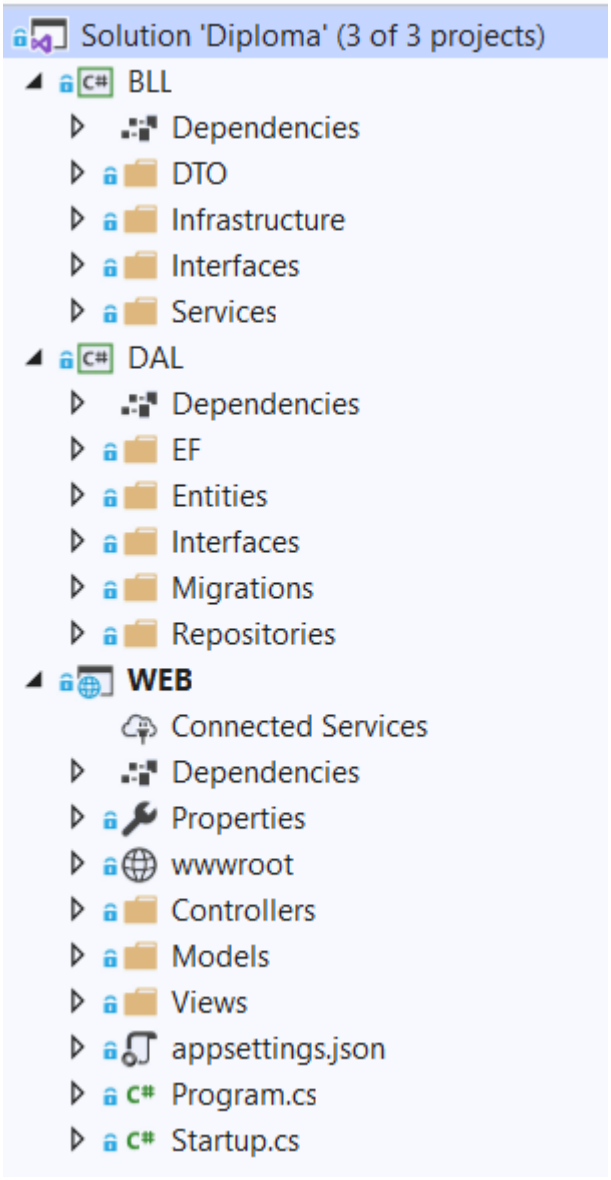

Рисунок 4.9 – Директорія системи

4.4 Створення компонентів системи

Для створення системи було використано два середовища: Visual Studio 2019 для розробки системи та MS SQL Server для створення концептуальної моделі бази даних на початку проектування. Більша частина дипломної роботи – створення форм для формування базових даних для подальшої роботи системи. Саме ця частина розроблялась в середовищі Visual Studio 2019.

4.4.1 Створення бази даних

Для початку роботи необхідно створити новий проект, для цього слід запустити програму Visual Studio 2019 та натиснути на “Create a new project”, після чого вибрати тип проекту ASP.NET Core Application та вказати шаблон.

Наступним кроком є створення необхідних класів, яке здійснюється при наведення та натисненні ПКМ на відповідну папку та виборі “Add” та типу Class. Class - це клас, що представлятиме сутність та її властивості, який пізніше буде таблицею в нашій базі даних.

Подібним чином відбувається додавання всіх всіх необхідних класів, для нашої бази даних(рисунок 4.10)

Рисунок 4.10 – Основні класи

Наступним кроком є створення контексту даних, яке відбувається за допомогою створення нового класу та успадковуванням у класу DbContext. В цьому класі створюються колекції об'єктів потрібних для створення БД та їх налаштування, рисунки 4.11 та 4.12.

```
76 references
public class ApplicationContext : DbContext
{
    5 references
    public virtual DbSet<User> Users { get; set; }
    5 references
    public virtual DbSet<AcademicTitle> AcademicTitles { get; set; }
    5 references
    public virtual DbSet<AcademicYear> AcademicYears { get; set; }
    5 references
    public virtual DbSet<BasicEducation> BasicEducations { get; set; }
    5 references
```

Рисунок 4.11 – Створення колекцій об’єктів

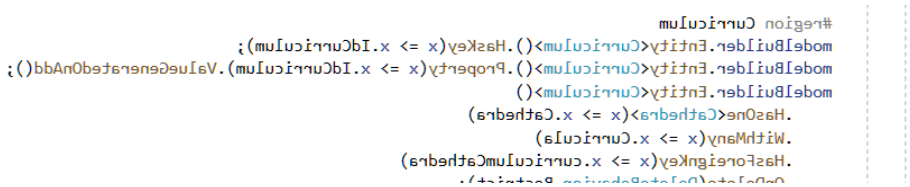

Рисунок 4.12 – Налаштування об’єктів

Після чого потрібно у Package Manager Console зробити міграцію бази даних та її оновлення за допомогою команд: add-migration MigrationName(Рисунок 4.13) та update-database(Рисунок 4.14). В результаті міграції створюється папка під назвою Migrations, яка містить файли зі створенням бази даних та її змінами, а також створюється сама база даних в MSSQL Server(Рисунок 4.15), яка відповідає раніше проектованій концептуальній моделі.

```
PM> add-migration example
Build started...
Build succeeded.
Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 3.1.3 initialized 'ApplicationContext' using provider 'Microsoft.EntityFrameworkCore.SqlServer' with options: None
To undo this action, use Remove-Migration.
```

Рисунок 4.13 – Міграція бази даних

```
PM> update-database
Build started...
Build succeeded.
Microsoft.EntityFrameworkCore.Infrastructure[10403]
      Entity Framework Core 3.1.3 initialized 'ApplicationContext' using provider 'Microsoft.EntityFrameworkCore.SqlServer' with options: None
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (20ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT 1
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (21ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT OBJECT_ID(N'[__EFMigrationsHistory]');
Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (2ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
```

Рисунок 4.14 – Оновлення бази даних

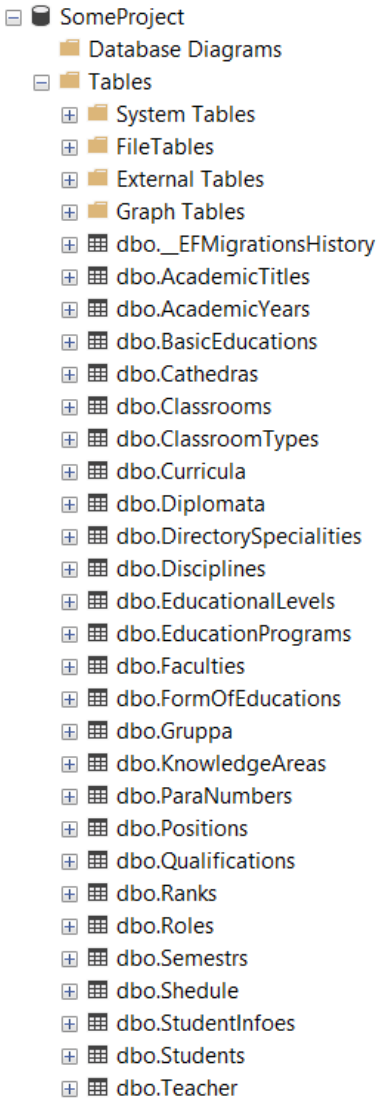

Рисунок 4.15 – Отримана база даних

Після цього переходимо до створення інтерфейсів та репозиторіїв для використання шаблону проектування Unit Of Work.

4.4.2 Написання сервісів для роботи з базою даних

Цей рівень є дуже важливим, оскільки він поєднує інші два рівні між собою, а саме рівень представлень та рівень доступу до даних тому, що рівень представлень не може на пряму доступатися до даних. В даному випадку цей рівень виступає посередником між іншими двома. Спочатку потрібно створити проміжкові об’єкти даних, які будуть пов’язуватися з відповідними об’єктами на інших двох рівнях(Рисунок 4.16).

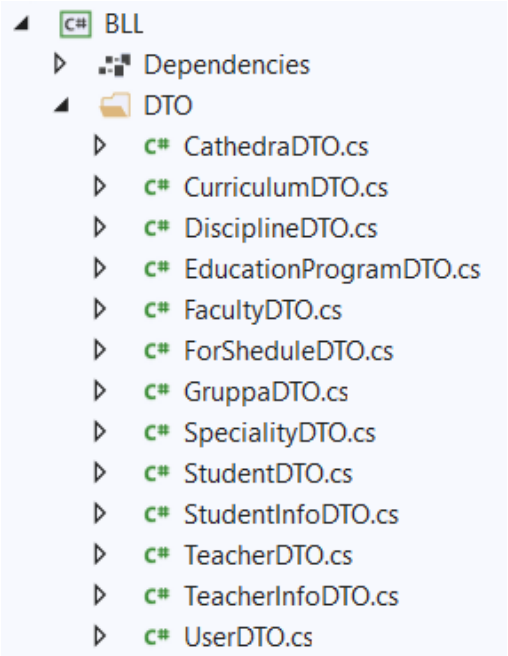

Рисунок 4.16 – ДТО моделі об’єктів

Дані класи також використовуються для об’єднання деяких об’єктів з основної БД. Після створення всіх потрібних ДТО моделей, переходимо до створення взаємодії між іншими рівнями за допомогою спеціальних сервісів. Для більшої гнучкості спочатку визначимо їх інтерфейси(Рисунок 4.17).

Також після створення інтерфейсів та їх сервісів потрібно буде за допомогою виклику services.AddTransient<InterfaceName, ServiceName>(); в методі ConfigureServices сказати, що система на місце об’єктів інтерфейсу InterfaceName, буде передавати екщемпляри класу ServiceName. Після додавання в ConfigureServices сервіси можна буде отримати та використовувати з будь-якої частини програми.

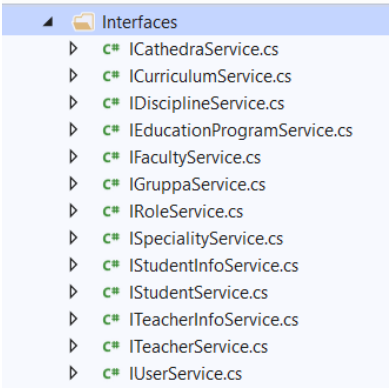

Рисунок 4.17 – Інтерфейси до сервісів

Для збереження реалізацій інтерфейсів визначимо наступну папку Services, в якій будуть міститися всі сервіси потрібні для повноцінного функціонування нашої системи(Рисунок 4.18).

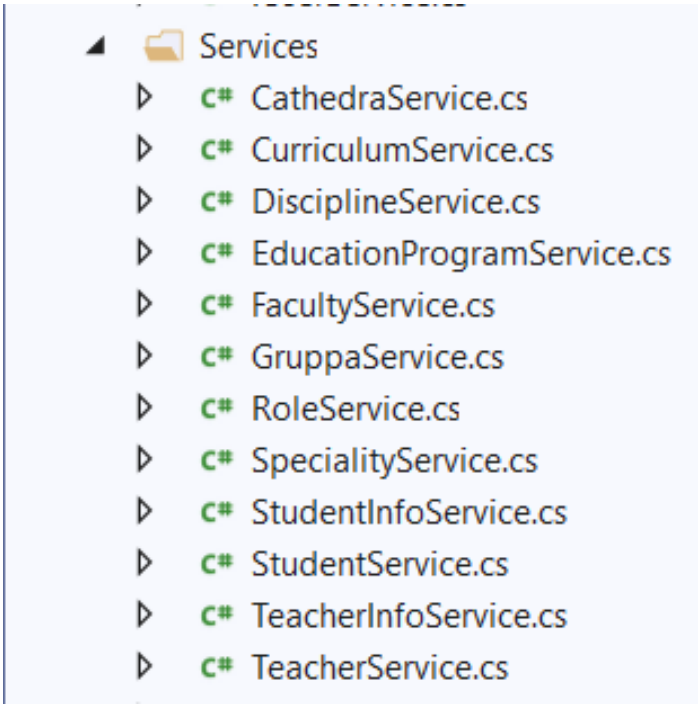

Рисунок 4.18 – Сервіси для роботи

4.4.3 Створення клієнтської частини

Клієнтська частина написана за допомогою платформи синтаксису

Razor, що поєднує синтаксис HTML та C#, а також моделей та контролерів. Даний рівень відповідає за взаємодію з користувачем.

Для переходу між сторінками використовувалися контролери, які обробляли запити та виконували певну дію(Рисунок 4.19).

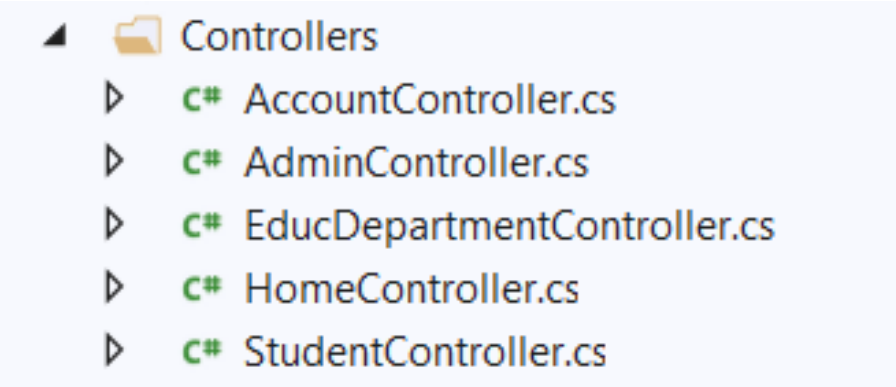

Рисунок 4.19 – Контролери на рівні представлень

Вони являються центральною частиною в архітектурі MVC. При отриманні запиту система маршрутизації обирає для опрацювання потрібний контролер та передає йому дані запиту. Контролер опрацьовує ці дані та посилає назад результат опрацювання. В ASP.NET Core контролер являється звичайним класом на мові C#, який успадковується від абстрактного базового класу Microsoft.AspNetCore.Mvc.Controller.

Одним з ключових компонентів архітектури MVC є моделі. Їх головним завданням є описати структуру та логіку використовуваних даних. Моделі слугують для поєднання рівня з рівнем бізнес логіки та заповнення даних, які потім будуть додані в нашу базу даних(Рисунок 4.20).

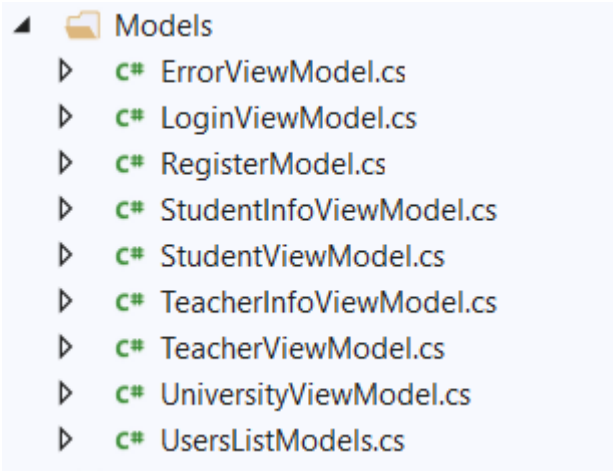

Рисунок 4.20 – Моделі рівня представлень

В ASP.NET Core MVC моделі можна розділити на за ступенем використання на декілька груп:

– Моделі, об’єкти яких зберігаються в спеціальних сховищах даних.

– Моделі, які для передачі даних представлення і навпаки, для отримання даних з представлення. Такі моделі називаються моделями представлень.

– Допоміжні моделі для проміжкових обчислень.

Хоча робота додатків MVC керується головним чином контролерами, то користувачу додаток доступний у вигляді представлення, яке і формує зовнішній вид застосунку. В ASP.NET Core MVC це файли з розширенням cshtml, які місять код користувацького інтерфейсу на мові HTML(Рисунок 4.21).

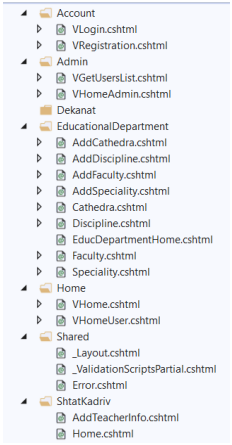

Рисунок 4.21 – Представлення

4.4 Висновки до розділу 4

При розробці програмного продукту важливим чинником є правильний вибір засобів реалізації застосунку, які впливають на час витрачений на розробку, якість, безпечність та швидкодію фінальної розробленої системи.

Засоби, які були обрані для програмної реалізації даного додатку забезпечили:

– ефективність розробки, використовуючи популярну мову C#;
– ефективність моделювання та гнучкість модифікації моделі бази даних з використанням EF Core;
– простоту інтеграції бази даних.

Для основи програмної системи було обрано такі засоби розробки, які дозволяють розроблювати високонавантажені асинхронні системи та одночасного доступу великої кількості користувачів.

5 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для забезпечення безвідмовної роботи системи треба дотримуватися основних вимог при інсталяції та рекомендацій щодо її використання, а також потрібне постійне та якісне з'єднання з мережею Internet для нормального використання системи та її роботи.

В цьому розділі наведено детальні системні вимоги та інструкцію щодо інсталяції програмної системи. Також у розділі наведено сценарії роботи користувача з системою.

5.1 Системні вимоги

Для використання розробленої програмної системи персональний комп'ютер повинен мати операційну систему MS Windows (XP, 7, 8, 8.1, 10), чи дистрибутив Linux (Ubuntu, Debian, BSD), Mac OS, крім того, на персональному комп'ютері повинні бути встановлений будь-який веб браузер.

Для запуску та експлуатації розробленої системи інсталяція не потрібна, достатньо перейти за посиланням, після цього система запуститься і в браузері з'явиться вкладка системи.

Якщо виникає потреба використання додатку для розробки, для виправлення недоліків чи покращення існуючого функціоналу, потрібно усі необхідні пакети для роботи застосунку. Їх всіх можна безкоштовно завантажити на офіційному сайті корпорації Microsoft.

5.2 Сценарії роботи користувача з системою

В даному розділі буде розгляну роботу з інтерфейсною частиною. Інтерфейсна частина додатку поділяється на вікна, кожне з яких має свою адресу у браузерному
рядку, навігація реалізована за допомогою простого меню.

Користувач взаємодіє з графічним інтерфейсом, що доступний як веб додаток. Оскільки авторизація при відвідуванні сайту є обов'язковою, то першим, що користувач бачить при відвідуванні веб-застосунку, вікно авторизації або ж реєстрації, якщо він не є зареєстрованим у даній системі (Рисунок 5.1). Також в нього є можливість відновлення паролю в разі його втрати. Авторизація відбувається за допомогою введення адреси електронної пошти та паролю, вказаних при реєстрації. Авторизація потрібна, оскільки система працює з даними, що мають обмежений доступ. На рисунку 5.1 зображено головне вікно програми.

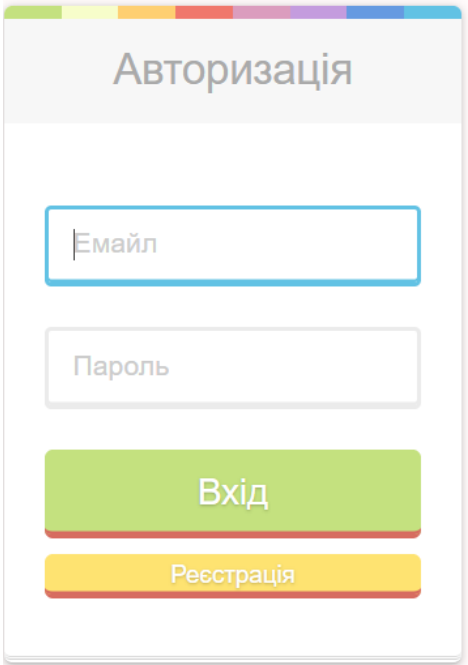

Рисунок 5.1 – Головне вікно системи при авторизації

Якщо ж користувач ще не зареєстрований, то він може перейти на сторінку реєстрації та створити собі обліковий запис. Сторінку реєстрації показано на рисунку 5.2.

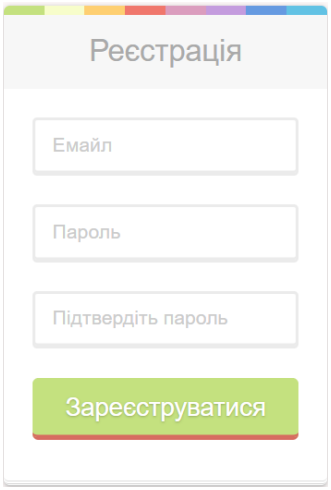

Рисунок 5.2 – Форма реєстрації нового користувача

Після авторизації користувач переходить на головну сторінку відповідно своєї ролі в даній системі, оскільки відділ кадрів та штат студентів або освітній відділ мають різні зобов'язання та можливості. Головну сторінку для навчального відділу зображено на рисунку 5.3.

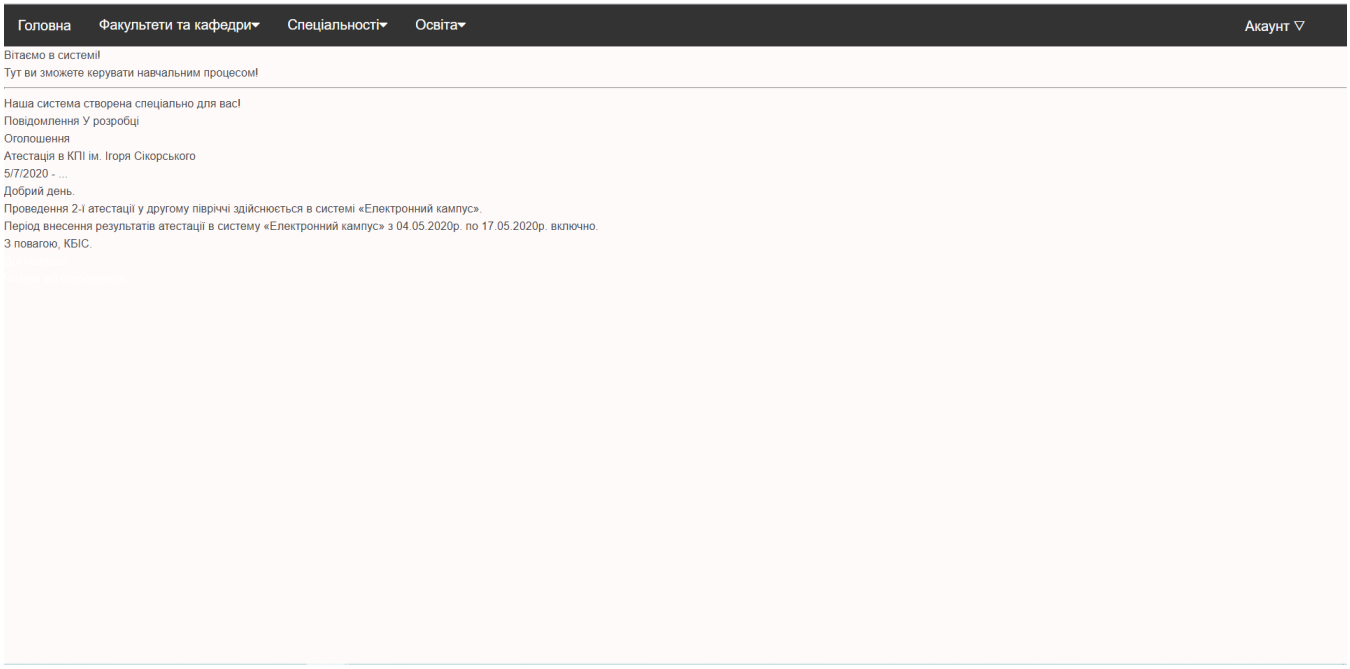

Рисунок 5.3 – Головна сторінка навчального відділу

Наступне вікно на рис. 5.4 дає нам зможу переглянути всі факультети в університеті.

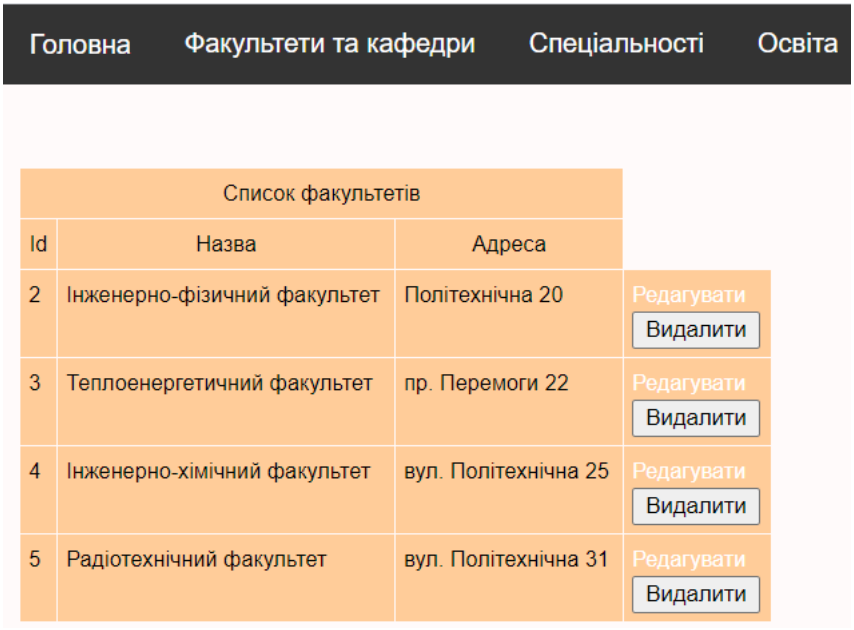

Рисунок 5.4 – Перегляд всіх факультетів

Третє вікно програми (рис. 5.5) дозволяє нам додавати кафедри в університеті та вказувати факультет, до якого вони належать.

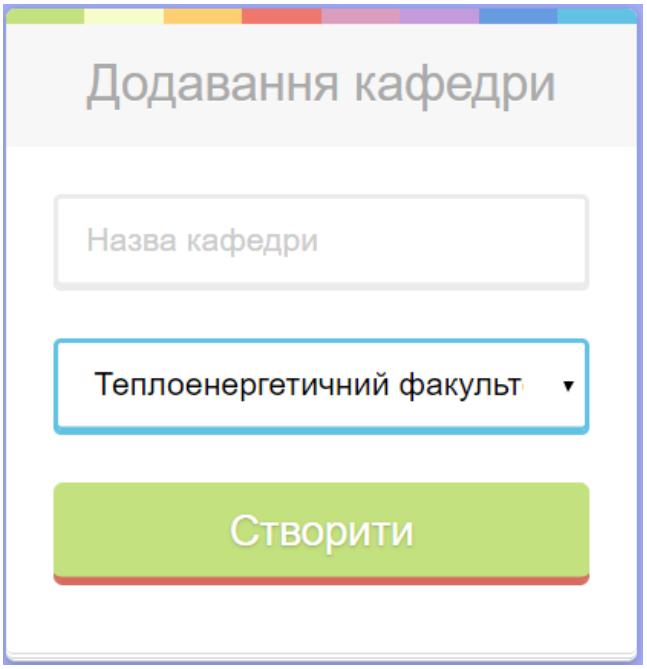

Рисунок 5.5 – Додавання кафедри

Також система дозволяє додавати та видаляти галузі знань(напрями підготовки). Це зображено на рисунку 5.6.

Список галузей		
Id	Назва	Шифр
1	Освіта	01
2	Культура і мистецтво	02
3	Гуманітарні науки	03
4	Інформаційні технології	12
5	Механічна інженерія	13

Рисунок 5.6 – Перелік галузей знань

ВИСНОВКИ

В ході виконання дипломної роботи було розроблено модуль формування базових даних для інформаційної системи автоматизованої підтримки навчальної діяльності кафедри. Розроблено архітектуру системи, яка включає набір спеціалізованих підсистем та базу даних. Сформована концептуальна модель бази даних на основі методики побудови реляційних баз даних. Обрано засоби розробки, до яких відносяться мова програмування C#, фреймворки EF Core, ASP.NET Core та Razor, які дозволяють зручно розроблювати та підтримувати систему, а також додавати нові компоненти. В якості СУБД обрано SSMS, що забезпечує надійну обробку даних. Розроблено програмне забезпечення та проведено його тестування, сформовано опис програмного продукту та методику роботи з ним для користувачів.

Потенційними користувачами системи є працівники вищих навчальних закладів, а також студенти.

Розроблений продукт може доповнюватися іншими більш складними спеціалізованими модулями. Завдяки використанню веб-інтерфейсу розроблений продукт не залежить від ОС користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ковтанюк Ю.С., Макарченко П.М., Дубровіна Л.А. Описування та організація зберігання електронних інформаційних ресурсів органів державної влади, органів місцевого самоврядування, підприємств, установ і організацій: методичні рекомендації / Укрдержархів України, УНДІАСД : Київ, 2010. – 30 с.
2. Тимчасове положення про організацію освітнього процесу в КПІ ім. Ігоря Сікорського: 5.4. Навчально-методичне забезпечення навчальних дисциплін.
3. Белянина И.Н., Богомаз И.В. Познавательные барьеры студентов ВУЗа и педагогические условия их преодоления / И.Н. Белянина, И.В. Богомаз // Вестник ТГПУ. – 2014. – №2. – С. 114-116.
4. Иан Грэхем. Объектно-ориентированные методы. Принципы и практика = Object-Oriented Methods: Principles & Practice. –3-е изд. – М.: «Вильямс», 2004. – С. 880.
5. Connolly R.: Fundamentals of Web Development / Pearson, 2017
6. ІТ-забезпечення діяльності іінноваційного університету: досвід українського вишу: монографія / А. В. Васильєв, В. О. Любчак, Ю. О. Зубань та ін. ; за заг. ред. проф. А. В. Васильєва. – Суми : Сумський державний університет, 2016. – 173с. ISBN 978-966-657-649-4.
7. Benjamins V.R. An Intelligent Brokering Service for Knowledge-Component Reuse on the World – Wide Web / Benjamins V.R. // Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management, June 2009, Чикаго, США , матеріали. — Ч.: W360, 2009 С. 125-129.
8. Разработка приложений MVC ASP.NET Core [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/develop-asp-net-core-mvc-apps.
9. Начало работы с EF Core [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/ef/core/get-started/?tabs=netcore-cli.

10. Основные принципы построения баз данных, проблемы хранения больших объемов информации [Електронний ресурс] – Режим доступу: https://sites.google.com/site/gosyvmkss12/bazy-dannyh/1-osnovnye-principy-postroenia-baz-dannyh-problemy-hranenia-bolsih-obemov-informacii.

11. Visual Studio 2019 – новая версия среды разработки от компании Microsoft [Електронний ресурс] – Режим доступу: https://info-comp.ru/programmirovanie/739-install-visual-studio-2019-community.html.

12. Общие сведения о платформе .NET Core [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/core/.

13. Антони Синтес. Освой самостоятельно объектно-ориентированное программирование за 21 день = Sams Teach Yourself Object-Oriented Programming in 21 Days. – М.: «Вильямс», 2002. – С. 672.

14. Торндайк Эдвард Ли. Принципы обучения, основанные на психологии. Пер. с англ. Е.А. Герье. Со вступит. Статьей Л.С.Выготского. Изд. 3-е. М., Работник просвещение, 1930.

15. Основи UML – проектування розподілених систем [Електронний ресурс] – Режим доступу: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?inpopup=true&id=44416

16. Мейер Д. «Теория реляционных баз данных» / Мейер Д. /: Пер. С англ. - М.: Мир, 1987, 608 с.

17. Введение в Entity Framework Core [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/entityframeworkcore/1.1.php

18. Введение в ASP.NET Core [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/aspnet5/1.1.php

19. Язык программирования C# 7 и платформы .NET и .NET Core / Э. Троелсен, Ф. Джепикс. – М.: ДМК Пресс; Спб.: Питер, 2017. – 1328 с.

20. Unit of Work [Електронний ресурс] – Режим доступу: https://martinfowler.com/eaaCatalog/unitOfWork.html

21. Многоуровневая архитектура [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/mvc5/23.5.php

22. Dependency injection [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-3.1

23. SQL Server Management Studio (SSMS) [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017

24. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. – М.: ДМК Пресс; Спб.: Питер, 2017. – 896 с.

25. Технологии проектирования баз данных / Осипов Д. Л. – М.: ДМК Пресс Питер, 2019. – 498 с.

26. Паттерны проектирования на платформе .Net / Сергей Тепляков – М.: ДМК Пресс Питер, 2015. – 320 с.

27. HTML и CSS. Разработка и создание веб-сайтов / Джон Дакетт – М.: «Эксмо», 2016. – 480 с.

28. Документация по C# [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/csharp/

29. Язык программирования C# 7 и платформы .NET и .NET Core / Эндрю Троелсен, Филипп Джепикс – М.: «Диалектика-Вильямс», 2019. – 1328 с.

30. Duckett J. HTML & CSS Design and Build Websites / J. Ducket. — Indianapolis: John Wiley & Sons, 2011.

ДОДАТОК А

Модуль формування базових даних інформаційної системи автоматизованої підтримки навчальної діяльності кафедри

Специфікація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61101_20Б

Аркушів 2

Київ – 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61 101_20Б 81-1	5. Опис.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61 101_20Б 12-1	WEB.csproj	Проект клієнтської частини продукту
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61 101_20Б 12-2	DAL.csproj BLL.csproj	Проекти серверної частини продукту
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61 101_20Б 13-1	Код_опис.docx	Опис програмного коду

ДОДАТОК Б

Модуль формування базових даних інформаційної системи автоматизованої підтримки навчальної діяльності кафедри

Лістинг програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61101_20Б

Аркушів 14

Київ – 2020

```
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using DAL.Entities;
using System.Security.Cryptography.X509Certificates;

namespace DAL.EF
{
   public class ApplicationContext : DbContext
   {
      public ApplicationContext(DbContextOptions connectionString) :
base(connectionString)
      {

      }


      public virtual DbSet<User> Users { get; set; }
      public virtual DbSet<Faculty> Faculties { get; set; }
      public virtual DbSet<Cathedra> Cathedra { get; set; }
      public virtual DbSet<Specialization> Specializations { get; set; }
      public virtual DbSet<KnowledgeArea> KnowledgeAreas { get; set; }
      public virtual DbSet<DirectorySpeciality> DirectorySpecialitys { get; set;
}
      public virtual DbSet<Role> Roles { get; set; }
      public virtual DbSet<Cycle> Cycles { get; set; }
      public virtual DbSet<UnderCycle> UnderCycles { get; set; }
      public virtual DbSet<Form_of_education> Form_Of_Educations { get; set;
}
      public virtual DbSet<FormDelivery> FormDeliveries { get; set; }
      public virtual DbSet<EducationalDegree> EducationalDegrees { get; set; }
      public virtual DbSet<Curriculum> Curriculums { get; set; }
      public virtual DbSet<PlanEducationalProcess> PlanEducationalProcesses
{ get; set; }

      //----------------
      public virtual DbSet<AcademicTitle> AcademicTitles { get; set; }
      public virtual DbSet<AcademicYear> AcademicYears { get; set; }
      public virtual DbSet<BasicEducation> BasicEducations { get; set; }
      public virtual DbSet<Cathedra> Cathedras { get; set; }
      public virtual DbSet<Classroom> Classrooms { get; set; }
      public virtual DbSet<ClassroomType> ClassroomTypes { get; set; }
```

```
        public virtual DbSet<Curriculum> Curricula { get; set; }
        public virtual DbSet<WeekDay> WeekDays { get; set; }
        public virtual DbSet<DirectorySpeciality> DirectorySpecialities { get; set;
}
        public virtual DbSet<Gruppa> Gruppas { get; set; }
        public virtual DbSet<ParaNumber> ParaNumbers { get; set; }
        public virtual DbSet<Position> Positions { get; set; }
        public virtual DbSet<Rank> Ranks { get; set; }
        public virtual DbSet<Semestr> Semestrs { get; set; }
        public virtual DbSet<Shedule> Shedules { get; set; }
        public virtual DbSet<Student> Students { get; set; }
        public virtual DbSet<StudentInfo> StudentInfoes { get; set; }
        public virtual DbSet<Teacher> Teachers { get; set; }
        public virtual DbSet<TeacherInfo> TeacherInfoes { get; set; }
        public virtual DbSet<TotalLoad> TotalLoads { get; set; }
        public virtual DbSet<Week> Weeks { get; set; }

        //----------------

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<User>().HasKey(x => x.ID_user);
            modelBuilder.Entity<Faculty>().HasKey(x => x.IdFaculty);
            modelBuilder.Entity<KnowledgeArea>().HasKey(x => x.IdArea);
            modelBuilder.Entity<DirectorySpeciality>().HasKey(x =>
x.IdSpeciality);
            modelBuilder.Entity<Cathedra>().HasKey(x => x.IdCathedra);
            modelBuilder.Entity<Specialization>().HasKey(x =>
x.ID_Specialization);
            modelBuilder.Entity<Cycle>().HasKey(x => x.IdCycle);
            modelBuilder.Entity<UnderCycle>().HasKey(x => x.IdUnderCycle);
            modelBuilder.Entity<Form_of_education>().HasKey(x =>
x.IdFormEducation);
            modelBuilder.Entity<FormDelivery>().HasKey(x => x.IdDelivery);
            modelBuilder.Entity<Curriculum>().HasKey(x => x.IDCurriculum);
            modelBuilder.Entity<PlanEducationalProcess>().HasKey(x =>
x.ID_PlanEducationalProcess);

            //-------------------------



            modelBuilder.Entity<AcademicTitle>().HasKey(x =>
x.IdAcademicTitle);
            modelBuilder.Entity<AcademicYear>().HasKey(x => x.IdYear);
            modelBuilder.Entity<ClassroomType>().HasKey(x => x.IdType);
            modelBuilder.Entity<Faculty>().HasKey(x => x.IdFaculty);
```

```
            modelBuilder.Entity<KnowledgeArea>().HasKey(x => x.IdArea);
            modelBuilder.Entity<ParaNumber>().HasKey(x => x.IdPara);
            modelBuilder.Entity<Position>().HasKey(x => x.IdPosition);
            modelBuilder.Entity<Rank>().HasKey(x => x.IdRank);
            modelBuilder.Entity<Semestr>().HasKey(x => x.IdSemestr);
            modelBuilder.Entity<StudentInfo>().HasKey(x => x.stundtInfoID);
            modelBuilder.Entity<Week>().HasKey(x => x.IdWeek);
            modelBuilder.Entity<WeekDay>().HasKey(x => x.IdDay);


            base.OnModelCreating(modelBuilder);
        }
    }
}

namespace DAL.Entities
{
    public class User
    {
        [Key]
        public int ID_user { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }

        [ForeignKey("Role")]
        public int ID_Role { get; set; }
        public virtual Role Role { get; set; }
    }
}
namespace DAL.Entities
{
    using System;
    using System.Collections.Generic;

    public partial class Faculty
    {
        [Key]
        public int IdFaculty { get; set; }
        public string facultyName { get; set; }
        public string facultyAddress { get; set; }
    }
}
namespace DAL.Repositories
{
    public class UnitOfWork : IUnitOfWork
    {
```

```
private ApplicationContext context;

private UserRepository userRepository;
private CathedraRepository cathedra;
private SpecializationRepository specialization;
private DirectorySpecialityRepository speciality;
private FacultyRepository faculty;
private KnowledgeAreaRepository area;
private RoleRepository rolesOFWork;
private FormDeliveryRepository formDeliverys;
private Form_of_educationRepository formEducation;
private CycleRepository cycles;
private UnderCycleRepository underCycles;
private EducationalDegreeRepository degrees;
private PlanEducationalProcessRepository planEducationalProcess;
private CurriculumRepository curriculums;

//-----
private AcademicTitleRepository academic;
private AcademicYearRepository year;
private BasicEducationRepository basic;
private ClassroomRepository classroom;
private ClassroomTypeRepository type;
private GruppaRepository gruppa;
private ParaNumberRepository para;
private PositionRepository position;
private RankRepository rank;
private SemestrRepository semestr;
private SheduleRepository shedule;
private WeekDayRepository day;
private WeekRepository week;
private RoleRepository role;
private StudentInfoRepository info;
private StudentRepository student;
private TeacherInfoRepository t_info;
private TeacherRepository teacher;
private TotalLoadRepository total;




public async Task SaveAsync()
{
   await context.SaveChangesAsync();
}
```

```
public UnitOfWork(DbContextOptions connectionString)
{
   context = new ApplicationContext(connectionString);
}

public IRepositoryMain<User, string> RUsers
{
   get
   {
      if (userRepository == null)
         userRepository = new UserRepository(context);
      return userRepository;
   }
}


public IRepositoryOnlyGet<Role> RRoles
{
   get
   {
      if (rolesOFWork == null)
         rolesOFWork = new RoleRepository(context);
      return rolesOFWork;
   }
}

public IRepositoryMain<DirectorySpeciality, string> RSpecialities
{
   get
   {
      if (speciality == null)
         speciality = new DirectorySpecialityRepository(context);
      return speciality;
   }
}

public IRepositoryMain<Faculty, string> RFaculties
{
   get
   {
      if (faculty == null)
         faculty = new FacultyRepository(context);
      return faculty;
   }
}
```

```
public IRepositoryMain<KnowledgeArea, string> RAreas
{
   get
   {
      if (area == null)
         area = new KnowledgeAreaRepository(context);
      return area;
   }
}

public IRepositoryMain<Cathedra, string> RCathedra
{
   get
   {
      if (cathedra == null)
         cathedra = new CathedraRepository(context);
      return cathedra;
   }
}

public IRepositoryMain<Specialization, string> RSpecialization
{
   get
   {
      if (specialization == null)
         specialization = new SpecializationRepository(context);
      return specialization;
   }
}

public IRepositoryMain<ClassroomType, string> RClassroomTypes
{
   get
   {
      if (type == null)
         type = new ClassroomTypeRepository(context);
      return type;
   }
}
public void Dispose()
{
   Dispose(true);
   GC.SuppressFinalize(this);
}

public async Task Save()
```

```
        {
            await context.SaveChangesAsync();
        }
    }
}

public class CathedraService : ICathedraService
    {
        private IUnitOfWork Database { get; set; }

        public CathedraService(IUnitOfWork uow)
        {
            Database = uow;
        }



        public IEnumerable<CathedraDTO> GetAll()
        {
            List<Cathedra> newList = Database.RCathedra.GetAll().ToList();
            List<CathedraDTO> newList_ = new List<CathedraDTO>();

            for (int i = 0; i < newList.Count(); ++i)
            {
                newList_.Add(new CathedraDTO
                {
                    Id_Cathedra = newList[i].IdCathedra,
                    cathedraName = newList[i].cathedraName,
                    facultyName = Database.RFaculties.GetAll().Where(x =>
x.IdFaculty == newList[i].ID_faculty).SingleOrDefault().facultyName
                });
            }

            return newList_;
        }

        public CathedraDTO FindCathedra(string name)
        {
            Cathedra item = Database.RCathedra.Get(name);
            CathedraDTO itemDTO = null;

            if (item != null)
            {
                itemDTO = new CathedraDTO();

                itemDTO.Id_Cathedra = item.IdCathedra;
```

```
            itemDTO.cathedraName = item.cathedraName;
            itemDTO.facultyName = Database.RCathedra.GetAll().Where(x =>
x.IdCathedra == item.IdCathedra).SingleOrDefault().cathedraName;
        }

        return itemDTO;
    }

    public async Task<OperationDetails> Create(CathedraDTO itemDTO)
    {
        Cathedra newItem = new Cathedra()
        {
            cathedraName = itemDTO.cathedraName,
            ID_faculty =
Database.RFaculties.Get(itemDTO.facultyName).IdFaculty
        };

        Database.RCathedra.Create(newItem);
        await Database.Save();

        return new OperationDetails(true, "Registration success", "");
    }
    public async Task<OperationDetails> Remove(int cathedraId)
    {
        string ct = Convert.ToString(cathedraId);
        Cathedra cathedra = Database.RCathedra.Get(ct);
        if (cathedra == null) throw new ValidationException("Cathedra is not
found", "");
        Database.RCathedra.Delete(ct);
        await Database.Save();
        return new OperationDetails(true, "Cathedra was deleted", "Cathedra");
    }

    public async Task<OperationDetails> ChangeInformation(CathedraDTO
cathedraDTO)
    {
        string id = Convert.ToString(cathedraDTO.Id_Cathedra);
        Cathedra cathedra = Database.RCathedra.Get(id);
        if (cathedra == null) throw new ValidationException("Cathedra is not
found", id);
        Cathedra newCathedra = new Cathedra();
        newCathedra.cathedraName = cathedraDTO.cathedraName;
        newCathedra.ID_faculty =
Database.RFaculties.Get(cathedraDTO.facultyName).IdFaculty;

        Database.RCathedra.Update(newCathedra);
```

```
            await Database.Save();
            return new OperationDetails(true, "Cathedra successfully changed",
"Cathedra");
        }
            }
```

ДОДАТОК В

Модуль формування базових даних інформаційної системи автоматизованої підтримки навчальної діяльності кафедри

Опис програмного модулю

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61101_20Б

Аркушів 8

Київ – 2020

АНОТАЦІЯ

У даному додатку міститься інформація, що описує програмний продукт для формування базових даних інформаційної системи кафедри. Даний програмний продукт дозволяє користувачеві заповнювати даними базу, видаляти їх та редагувати.

Програма була розроблена у середовищі Visual Studio 2019 мовою програмування C# з використанням фреймворків EF Core, ASP.NET Core та Razor.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Даний програмний продукт було розроблено в середовищі Visual Studio 2019 мовою програмування C# з використанням фреймворків EF Core, ASP.NET Core та Razor.

Програма призначена для адміністрації, якій необхідно швидко та зручно вводити дані про навчальний процес кафедри та додаткову інформацію. Принцип роботи програми побудовано на заповненні даними таблиці, інформація з яких буде використовуватися в подальшому.

До основних переваг програми можна віднести швидкість роботи на відносно простий для користувача інтерфейс.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

При створенні програмного продукту була поставлена задача зберігання, заповнення та відображення користувачем великої кількості даних. Тому потрібно було спроектувати таку модель бази даних, щоб було зручно нею користуватися, вносити дані та швидко їх отримувати.\

Програмний продукт містить три проекти, які формують його трьохрівневу архітектуру, такі як:

- DAL – data access layer – рівень доступу даних;
- BLL – business logic layer – рівень бізнес логіки;
- WEB – presentation layer – рівень представлень.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Створена програма складається з декількох частин. Першою частиною ж є проект DAL.csproj, в якому створюється майбутня база даних та виконуються операції з нею. Другою частиною є проект BLL для побудови бізнес логіки нашого застосунку. До інтерфейсу користувача входить третя частина, WEB.csproj, з усіма сторінками для взаємодії з користувачем.

Робота програми побудована таким чином, що перейшовши на сайт та авторизувавшись, користувач взаємодіє з сервісом залежно від своєї ролі. Оскільки така система є громіздкою, то ролей в ній повинно бути багато і кожна повинна мати свій функціонал. В даному випадку користувач взаємодіятиме із застосунком як працівник навчального відділу, який може заповнювати інформацією базу. Ці дані будуть використовуватися іншими ролями.

З отриманими даними користувач також може взаємодіяти, для того щоб їх редагувати або видаляти.

ВИКОРИСТУВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для створення даної програми було використано середовище розробки Visual Studio 201 та мова програмування C# з використанням фреймворків EF Core, ASP.NET Core та Razor.

Для створення та редагування концептуальної моделі було використано MS SQL Server Management Studio.

Для запуску даної програми користувачу необхідно мати встановлений на своєму комп’ютері будь-який веб-браузер.

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними є:

- Інформація про кафедри, рівні та ступені освіти, спеціалізації та ін.

Вихідними даними є:

- виведення інформації з бази даних.

ДОДАТОК Г

Модуль формування базових даних інформаційної системи автоматизованої підтримки навчальної діяльності кафедри

Апробація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР61101_20Б

Аркушів 4

Київ – 2020

Було подано тези на наукову конференцію з автоматичного управління присвяченої Дню космонавтики. Тези надруковано в першій секції на сторінці 9. Текст тез наведено нижче.

Міністерство освіти і науки України

Херсонський національний технічний університет

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Вінницький національний медичний університет
ім. М. І. Пирогова

Луцький національний технічний університет

Вінницький національний технічний університет

Кременчуцький національний технічний університет
ім. Михайла Остроградського

Сумський державний університет

Херсонський державний аграрно-економічний університет

Матеріали VIII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених з автоматичного управління

присвячена Дню космонавтики

08 – 10 квітня 2020р.
Херсон

ЗМІСТ

СЕКЦІЯ «АВТОМАТИЗОВАНЕ УПРАВЛІННЯ ТЕХНОЛОГІЧНИМИ ПРОЦЕСАМИ»

СЕКЦІЯ «МОДЕЛЮВАННЯ ТА ОПТИМІЗАЦІЯ СИСТЕМ УПРАВЛІННЯ»

УДК 004.414.38

О.М. Белень, А.В. Дмитрук, О.М. Шушура
НТУУ «КПІ ім. Ігоря Сікорського»
alexandrbelen@gmail.com

АВТОМАТИЗАЦІЯ УПРАВЛІННЯ НАВЧАЛЬНОЮ ДІЯЛЬНІСТЮ УНІВЕРСИТЕТУ НА БАЗІ ASP.NET Core 3.0 ТА ANGULAR 9

Інформаційні технології стали невід'ємною частиною сучасного життя. Вони назавжди змінили світ бізнесу, культури, освіти, виробництва. Використання інформаційних технологій дозволило суттєво збільшити ефективність навчального процесу. Водночас постала проблема необхідності обробки великої кількості інформації, а отже, створення таких систем, що будуть полегшувати управління навчальним процесом. Такий напрям, як інформатизація освіти, має місце в усіх без винятку національних програмах руху до інформаційного суспільства.

Навчальний процес у вищому навчальному закладі достатньо важкий, і його ефективна організація потребує значних зусиль. У сучасних університетах автоматизація відбувається у двох основних взаємопов'язаних напрямках.

Перший напрям – автоматизація освітнього процесу, використання сучасних інформаційних технологій для модернізації педагогічного процесу (дистанційне навчання, електронне та всепроникне навчання).

Другий напрям – автоматизація системи університетського менеджменту шляхом розробки інформаційних технологій для бізнес-процесів сучасного університету.

В даній роботі розглядається комплексний підхід, який враховує обидва вказані напрямки. Огляд існуючих інформаційних систем в цій галузі дозволив виділити наступні принципи їх побудови:

- використання даних із загального сховища даних (інтегрована БД) з розмежуванням прав доступу на рівні користувачів і окремих додатків (окремих показників);
- використання загальних довідників;
- обмін інформацією між підсистемами на основі єдиного інформаційного середовища.

Проектування інтегрованої інформаційної системи повинне розглядатися крізь призму основних функцій управління, основних напрямів діяльності університету та його внутрішньої структури, що формується залежно від виконуваних завдань. Система повинна справлятися із значним навантаженням, одночасно обробляти запити користувачів та мати зручний і інтуїтивно зрозумілий інтерфейс.

Для створення клієнтської частини порталу обрані такі технології як Angular 9–фреймовк на основі паттерна MVC,інтеграція з ASP.NET Core 3.0, запити та обробка даних.

Основою серверної частини є крос-платформна система ASP.NET Core 3.0. Вибір даних технологій зумовило те, що дана система є складною в проектуванні та підтримці, а даний фреймворк призначений саме для таких випадків.

Використання наведених сучасних технологій розробки інформаційних систем дозволяє врахувати всі суттєві вимоги до управління навчальним процесом та забезпечити надійність і зручність застосування програмного продукту.

ЛІТЕРАТУРА:

1. Методологічні основи створення впровадження і розвитку інтегрованої інформаційної системи управління університетом. 2015 URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/47881/3/integrated_management_system_ukr.pdf.
2. Бибо Б. jQuery. Подробное руководство по продвинутому JavaScript. М.: Символ-Плюс, 2011.
3. ASP.NET Core. URL: https://docs.microsoft.com/en-us/aspnet/?view=aspnetcore-3.1.